D2.5

The OPTIMAI architecture specifications – 2nd version

30 June 2022

OPTIMAI

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 958264

DOCUMENT SUMMARY INFORMATION

Grant Agreement No	958264	**Acronym**	OPTIMAI
Full Title	Optimizing Manufacturing Processes through Artificial Intelligence and Virtualization		
Start Date	01/01/2021	**Duration**	36 months
Deliverable	D2.5: The OPTIMAI architecture specifications – 2nd version		
Work Package	WP2 – User requirements, Technical Specifications and Use case analysis		
Nature	Report	**Dissemination Level**	Public
Lead Beneficiary	FORTH		
Authors	Konstantinos C. Apostolakis (FORTH) George Margetis (FORTH)		
Co-authors	Stefania Stamou (FORTH) Nikolaos Dimitriou (CERTH) Christina Tsita (CERTH) Walter Domenico Vergara (ENG) Manfredi Giuseppe Pistone (ENG) George Bogdos (FINT) George Alexiou (FINT) Andreas Böttinger (EVT) Ali Sadr (EVT) Elpiniki Papageorgiou (UTH) Theodosis Theodosiou (UTH) Andrea Gomez (UNIMET) Clara Valero (UPV) Antonio Zanesco (YBQ) Greg Tinker (MTCL) Fernando Ubis (VIS) Agata Gurzawska (TRI)		
Reviewers	Walter Domenico Vergara (ENG) Manfredi Giuseppe Pistone (ENG) Elisa Rossi (ENG) Sabrina Verardi (ENG)		

DISCLAIMER

The OPTIMAI Project receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 958264. The sole responsibility for the content of this document lies with the authors. It does not necessarily reflect the opinion of the European Union. The European Commission is not responsible for any use that may be made of the information contained herein.

DOCUMENT HISTORY

Version	Date	Changes	Contributor(s)
V0.1	05/05/2022	Initial deliverable structure	Konstantinos C. Apostolakis (FORTH)
V0.2	13/05/2022	Refinements to Sections 1, 3, Updates to the Executive Summary and Functional descriptions of intact components in Section 4	Konstantinos C. Apostolakis (FORTH) George Margetis (FORTH) Stefania Stamou (FORTH)
V0.3	18/05/2022	Inputs integration by YBQ	Antonio Zanesco (YBQ)
V0.4	20/05/2022	Inputs integration by ENG, UTH	Walter Domenico Vergara (ENG) Manfredi Giuseppe Pistone (ENG) Elpiniki Papageorgiou (UTH) Theodosis Theodosiou (UTH)
V0.5	23/05/2022	Inputs integration by EVT	Andreas Böttinger (EVT) Ali Sadr (EVT)
V0.6	30/05/2022	Inputs integration by CERTH, UNIMET	Nikolaos Dimitriou (CERTH) Christina Tsita (CERTH) Andrea Gomez (UNIMET)
V0.7	03/06/2022	Refinements to Information and Deployment Views (Sections 4 & 5)	Konstantinos C. Apostolakis (FORTH) George Margetis (FORTH) Stefania Stamou (FORTH)
V0.8	06/06/2022	Inputs integration by FINT	George Bogdos (FINT) George Alexiou (FINT)
V0.9	15/06/2022	Integration of inputs from W23 architecture workshop	George Bogdos (FINT) George Alexiou (FINT) Clara Valero (UPV) Greg Tinker (MTCL) Fernando Ubis (VIS) Agata Gurzawska (TRI)
V1.0	21/06/2022	Final version for internal review	Konstantinos C. Apostolakis (FORTH)

V1.1	27/06/2022	Internal review	Walter Domenico Vergara (ENG) Manfredi Giuseppe Pistone (ENG) Elisa Rossi (ENG) Sabrina Verardi (ENG)
V1.2	28/06/2022	First revisions after internal review & Final Version for Quality & Security Review	Konstantinos C. Apostolakis (FORTH)

PROJECT PARTNERS

Logo	Partner	Country	Short name
	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS	Greece	**CERTH**
	FINT FUTURE INTELLINGENCE LIMITED	Cyprus	**FINT**
	IDRYMA TECHNOLOGIAS KAI EREVNAS	Greece	**FORTH**
	EVT EYE VISION TECHNOLOGY GMBH	Germany	**EVT**
	VISUAL COMPONENTS OY	Finland	**VIS**
	YOUBIQUO SRL	Italy	**YBQ**
	PANEPISTIMIO THESSALIAS	Greece	**UTH**
	ENGINEERING – INGEGNERIA INFORMATICA SPA	Italy	**ENG**
	UNIMETRIK SA	Spain	**UNIMET**
	UNIVERSITAT POLITECNICA DE VALENCIA	Spain	**UPV**
	Carr Communications Limited	Ireland	**CARR**
	UNIVERSIDAD AUTONOMA DE BARCELONA	Spain	**UAB**
	TRILATERAL RESEARCH LIMITED	Ireland	**TRI**
	KLEEMANN HELLAS –INDUSTRIAL COMMERCIAL SOCIETE ANONYME FOR MECHANICAL CONSTRUCTION SA	GREECE	**KLEE**
	TELEVES SA	Spain	**TVES**
	MICROCHIP TECHNOLOGY CALDICOT LIMITED	United Kingdom	**MTCL**

LIST OF ABBREVIATIONS

Abbreviation	Definition
AI	Artificial Intelligence
AIPMC	AI-based Production Monitoring Component
API	Application Programming Interface
AR	Augmented Reality
BaaS	Blockchain-as-a-Service
BOM	Bill of Materials
CAI	Conversational Agent Interface
CEP	Complex Event Processing
CPPS	Cyber-Physical Production System
DIDA	Digital Industry Data Analytics
DML	Dedicated Manufacturing Line
DoA	Description of Action
DSS	Decision Support System
DT	Digital Twin
EbD	Ethics by Design
ERP	Enterprise Resource Planning
ETFA	IEEE Annual Conference on Emerging Technologies and Factory Automation
EVM	Ethereum Virtual Machine
F4I	FIWARE4Industry
FMS	Flexible Manufacturing System
FPGA	Field-Programmable Gate Array
GA	Grant Agreement
GAN	Generative Adversarial Network
GDTA	Generic Digital Twin Architecture
GE	Generic Enabler
HDFS	Hadoop Distributed File System
HMI	Human-Machine Interface
HITL	Human-in-the-Loop
HUD	Heads-Up Display

I4.0	Industry 4.0
IaaS	Infrastructure-as-a-Service
ICT	Information & Communication Technologies
IEC	International Electrotechnical Commission
IFD	Information Flow Diagram
IIRA	Industrial Internet Reference Architecture
(I)IoT	(Industrial) Internet of Things
IPFS	InterPlanetary File System
ISO	International Organization for Standardization
IVRA	Industrial Value Chain Reference Architecture
LASFA	LASIM Smart Factory
LASIM	Laboratory for handling, assembly and pneumatics (Acronym in Slovene)
LCtD	Legal Compliance through Design
LDS	Loosely-Defined Standards
LIDAR	Light Detection And Ranging
LSTM	Long Short Term Memory
MAIAR	optiMAI Augmented Reality
MES	Manufacturing Execution Systems
MQTT	Message Queuing Telemetry Transport
NGSI	Next Generation Service Interfaces
OMIDES	Operator-Machine Interaction & Decision Support
OPC UA	OPC (Foundation) Unified Architecture
OT	Operation Technologies
PaaS	Platform-as-a-Service
PC	Personal Computer
PCE	Production Control Engineer
PoE	Power over Ethernet
PLO	Product Line Operator
QCS	Quality Control Sensors
R&D	Research & Development
RA	Reference Architecture
RAMI	Reference Architectural Model Industrie

RBAC	Role-Based Access Control
REST	REpresentational State Transfer
ROI	Return on Investment
SaaS	Software-as-a-Service
SC	Smart Contract
SCADA	Supervisory Control and Data Acquisition
SCI 4.0	Standardization Council Industrie 4.0
SITAM	Stuttgart IT Architecture for Manufacturing
SOA	Service-Oriented Architecture
SoftSensor	Software Sensor
SoS	System-of-Systems
SPA	Single Page Application
SSO	Single Sign-On
ToF	Time of Flight
UML	Unified Modelling Language

Executive Summary

This report corresponds to the culmination of the system specification and architecture design activities undertaken over the first 18 months of the OPTIMAI project lifetime, and represents the final description of the OPTIMAI smart manufacturing solution architecture. The document thoroughly describes the updated version of the envisioned solutions' ecosystem, based on the elicited stakeholders' requirements and use case scenario definitions that preceded it.

The revised (and final) version of the OPTIMAI architectural stack is a result of a second iteration of the three-step architecture design methodology, which followed the original definition derived from exercising the approach over months 1-12. In this follow-up procedure, technology exploration has considered the current technological state regarding the development of key components in the OPTIMAI architecture, which have been reported in project deliverables from WP3 and WP6. A second top-down design approach was then carried out, so as to identify placement of potential new components and subsystems while maintaining the original vertical segmentation of the architecture, so as to align components to the specified needs and requirements of the end-users. After conclusion of this process, a total of 38 functional blocks were identified (including former, modified and new architectural blocks in relation to D2.4). Each component was then elaborated by project partners responsible for their implementation through a bottom-up functional specification, using a revised component refinement template. This template enabled partners to specify functional, information and deployment characteristics of their components, essentially providing a baseline for the re-specification of the relevant views of the architecture (Functional, Information, Deployment). Affirmation of the new architecture was carried out via two online workshops, where consensus was reached among the Consortium members, and preparation of the present report was greenlit.

It is therefore accepted that, where specified, the outcomes described in this report supersede the results of the initial architecture definition in D2.4, thus complementing the previous exploratory and design activities (both top-down and bottom-up) with new gathered evidence and elaborating on the various components with additional information collected in the form of online workshops with participation of all OPTIMAI partners. We hence also proceed to re-align the revised architecture to leading Industrie (I)4.0 reference frameworks (RAMI 4.0 and IIRA), so as to substantiate OPTIMAI as an I4.0-compliant approach to zero-defect manufacturing.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1 Introduction

This deliverable culminates the OPTIMAI consortium's overall efforts at defining a modular, service-oriented architecture for the OPTIMAI proposed solutions. The document, which is an updated version of D2.4 "The OPTIMAI Architecture specifications – 1st version", aims to serve as a reference guide for the consortium partners engaged in the technical implementation of the project, and are responsible for the development of the components that will comprise the final architecture to be deployed at the pilot sites. The components, information flows and topological considerations described in this deliverable are the result of a combination of the entirety of activities undertaken in the context of Task 2.3 "System specifications and architecture", both those that produced the contents reported in D2.4, as well as those undertaken after, which in some cases shall supersede the reported outputs of the 1st version architecture definition.

The first version of the OPTIMAI architecture described at M12 presented a comprehensive overview of the technical specifications, along with a first version of the functional viewpoint of the architecture aimed at representing the functional blocks comprising the architectural model. The functional view specified is presented in Figure 1, below.

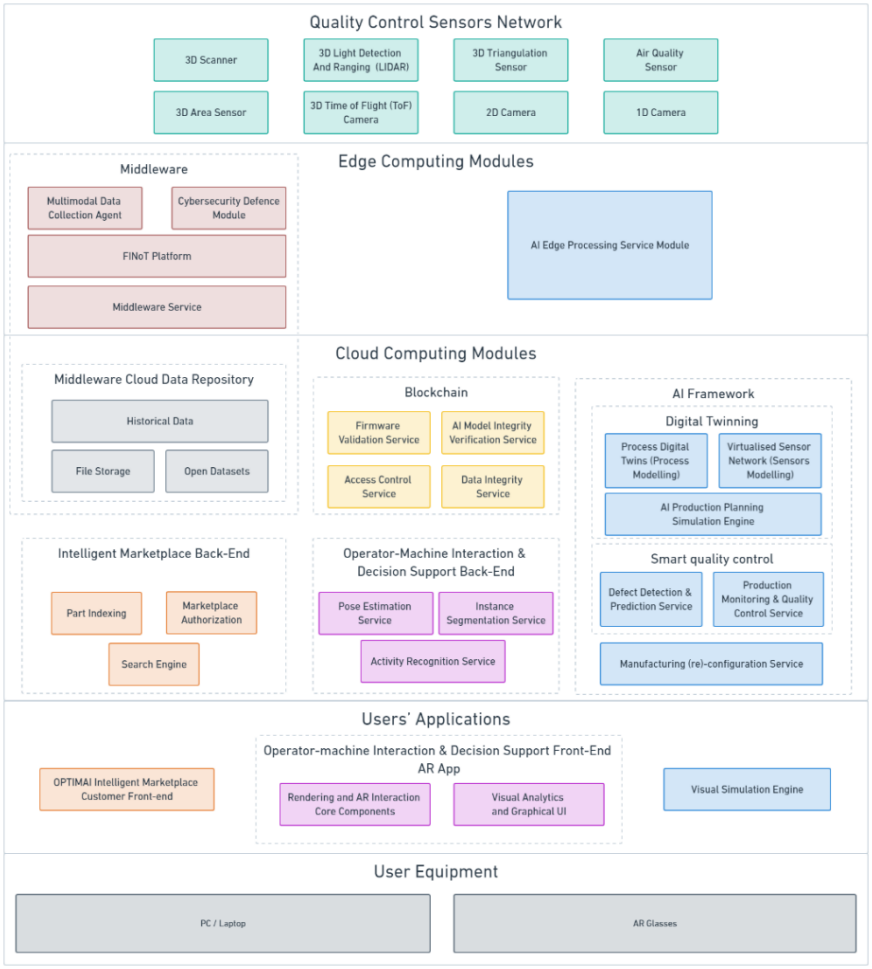


Figure 1: OPTIMAI overall functional architecture component diagram (retrieved from D2.4, M12).

The early specification of component responsibilities and roles was complemented by high-level overview of the interconnections between the specified functional architecture components to establish a base for the foreseen interdependencies of each functional block with other components. This has been reflected in Figure 2.

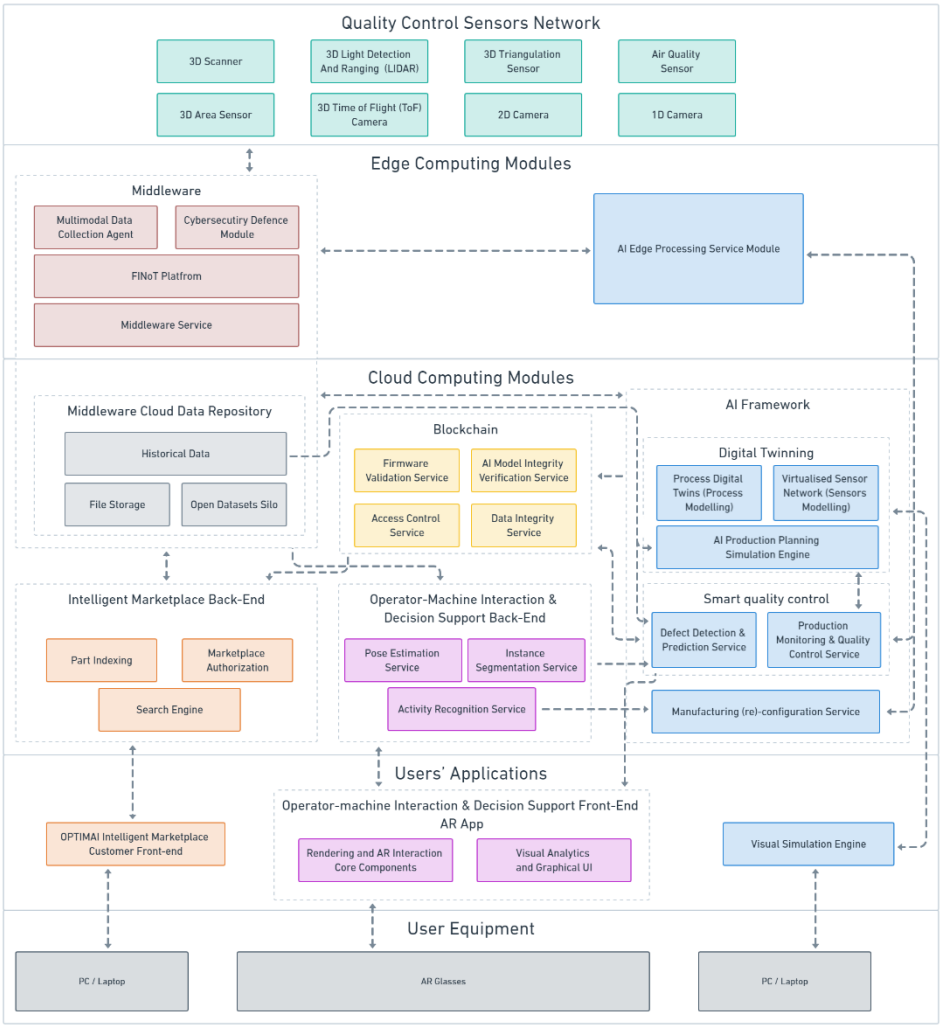


Figure 2: OPTIMAI architectural framework with information flows (retrieved from D2.4, M12).

Finally, deployment of the system was addressed by means of the framework of the Industry 4.0 smart factory [22], albeit extended to include also edge computing technologies, as shown in Figure 3.

The purpose of this deliverable is to further elaborate on the proposed framework, providing updates to the roles and responsibilities of the identified architectural building blocks as well as their interfaces and deployment characteristics. Essentially, this report describes the final system architecture resulting from 18 months of project implementation and architecture refinement activities. Therefore, through the contents of this deliverable, the integrated framework architecture is updated and finalized, encapsulating knowledge obtained through months 12-18 of the project and leading up to the concrete system description and elaboration for the deployment of the system at the OPTIMAI pilot sites.

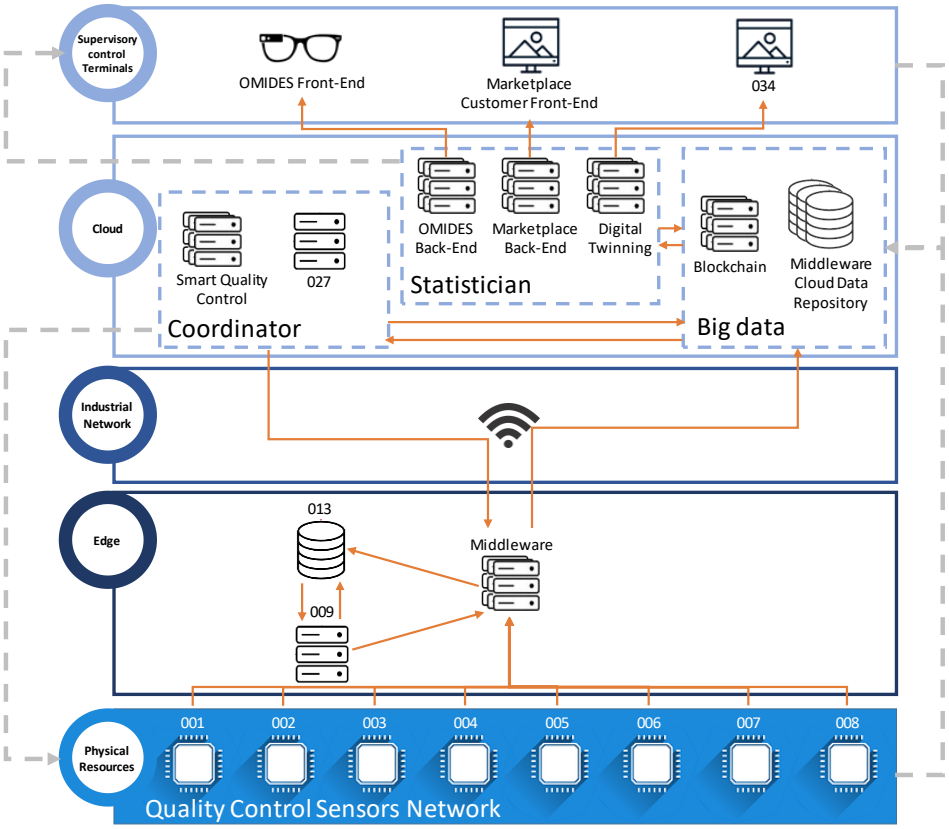


Figure 3: OPTIMAI topological definition (retrieved from D2.4, M12).

It is accepted that for the remainder of the project, the OPTIMAI architecture viewpoints as defined in the present document take precedence over the definition of the same architectural viewpoint as specified in D2.4.

The present document is structured as follows:

- Section 2 contains no updates since the D2.4 version.
- Section 3 presents an overview of the steps followed for the refinement of the architecture functional components and their interdependencies, following the D2,4-established architectural approach.
- Section 4 describes the functional viewpoint of the OPTIMAI architecture, presenting a detailed description of the architectural components and subsystems, and offering insight into the mappings drawn to the most prominent reference architecture models and guiding principles.
- Section 5 presents the information viewpoint of the OPTIMAI architecture, elaborating on the way information flows through the architectural components (presented at both a high-level overview and further targeted at indicative scenarios described in the identified OPTIMAI use cases).
- Section 6 presents the OPTIMAI solution's deployment over physical and virtualised resources, as well as the topological map based on the smart factory framework, which it extends to accommodate the foreseen edge computing capabilities.

Finally, in Section 7, the key points of the document are summarized.

1.1 Mapping of project outputs

The purpose of this section is to map OPTIMAI Grant Agreement (GA) commitments, both within the formal Deliverable and Task description, against the project's respective outputs and work performed. This mapping is presented in Table 1, below.

Table 1: Adherence to OPTIMAI's GA Deliverable & Tasks Descriptions

OPTIMAI Task	Respective Document Chapters	Justification
T2.3: System specifications and architecture *"In this scope, the definition of specifications for the various components of the system and the whole system as an entity will be attempted, so as to fulfil the existing and future demands of stakeholders".*	Section 4 - OPTIMAI Architecture - Functional view	The detailed description of the final functional blocks and subsystems underpinning the OPTIMAI stakeholders' requirements are described in Section 4 of this document.
T2.3: System specifications and architecture *"Based on the analysis of user needs and scenarios, conducted in T2.1 and T2.4, system specifications will be defined according to existing standards activities in order to address interoperability and security requirements for each module".*	Section 2 - Reference Architecture models	A comprehensive overview of standards and industry-led reference architectural models is provided in Section 2, which is firmly elaborated in D2.4 (Void in this report).
	Section 4.2 - OPTIMAI alignment to standards-led reference architectures	Section 4.2 details the mappings and parallels drawn between the refined OPTIMAI functional architecture perspective and two leading standardisation initiatives referring to the system as both an Industry 4.0 solution and an Industrial Internet of Things deployment.
T2.3: System specifications and architecture *"In addition, this task will analyse the requirements and recommendations of the*	Section 3.4 - Ethical compliance – Compliance through Design	Following the description of the architecture definition approach taken in the first year of the project, this Section presents key

OPTIMAI solution resulted from WP9 regarding the ethics and regulatory framework in order to define an architecture that reflects the concept of "security and privacy by design"".		principles followed to adhere to an Ethics by Design and Legal Compliance through Design approach (Void in this report, no changes since D2.4).
T2.3: System specifications and architecture *"Effort will be put system modularity so as to address diversity in equipment and resources, thus allowing for future changes, updates and upgrades."*	Section 6 - OPTIMAI Architecture - Deployment view	The final description of the foreseen tangible infrastructures necessary for supporting deployment of the system components in service to the project use cases is described in Section 6.
T2.3: System specifications and architecture *"The design of every architectural module will have to take into account all relevant and important elements like system requirements, risk factors, software issues, communication elements, safety issues, hardware requirements and specific application requirements".*	Section 3 - Architecture refinement approach	This Section covers the approach followed for defining the final version of the architecture described in this document.
	Section 4 - OPTIMAI Architecture - Functional view	This Section describes the responsibilities, roles and foreseen dependencies of the OPTIMAI functional blocks, driven by the elicited functional and non-functional requirements. The perspective of standards' bodies with respect to the OPTIMAI use cases is also taken into consideration.
	Section 5 - OPTIMAI Architecture - Information view	This Section describes the information flow within OPTIMAI, with a specific aim at demonstrating how information flows through the system for specific tasks identified in the project use cases.

	Section 6 - OPTIMAI Architecture - Deployment view	This Section elaborates on the topology for the smart factory solution proposed, with implications on hardware and software requirements, along with a view on the necessary applications' end-user terminals to target.

1.2 Updates since the initial deliverable version

It is accepted that, where specified, the outcomes described in this report supersede the results of the initial architecture definition (D2.4). Hence, the following Sections complement the previous exploratory and design activities (both bottom-up and top-down) with newly gathered evidences, elaborating on the various components with updated information collected in the form of online workshops (with participation of all OPTIMAI partners) and following careful examination of project deliverables. For this reason, to avoid repetition, whenever information remains constant across the two deliverables (D2.4 and D2.5), it is accepted that the content in D2.4 hold true. In these cases, the corresponding Sections in this document will be marked as *'Void'*. Whenever information from D2.4 is updated, but specific paragraphs from that report: (i) hold true; and (ii) are necessary to provide context for the updated information, those paragraphs will be included in the following manner:

> *Paragraph content.*
>
> Section **number**, D2.4, M12.

Whenever minor details in those descriptions are updated for this present version, those details will be highlighted in **bold**.

2 Reference Architecture models

Void (the contents provided in Section 2 of D2.4 apply).

2.1 Definitions and conventions used

Void (the contents provided in Section 2.1 of D2.4 apply).

2.2 Reference architectures for Industry 4.0

Void (the contents provided in Section 2.2 of D2.4 apply).

2.3 OPTIMAI reference implementation model

Void (the contents provided in Section 2.3 of D2.4 apply).

3 Architecture refinement approach

This Section presents the second iteration of the architectural design approach, as originally specified in D2.4. For the sake of completeness, the sequence of implemented actions over the course of a single iteration is summarized in Figure 4.

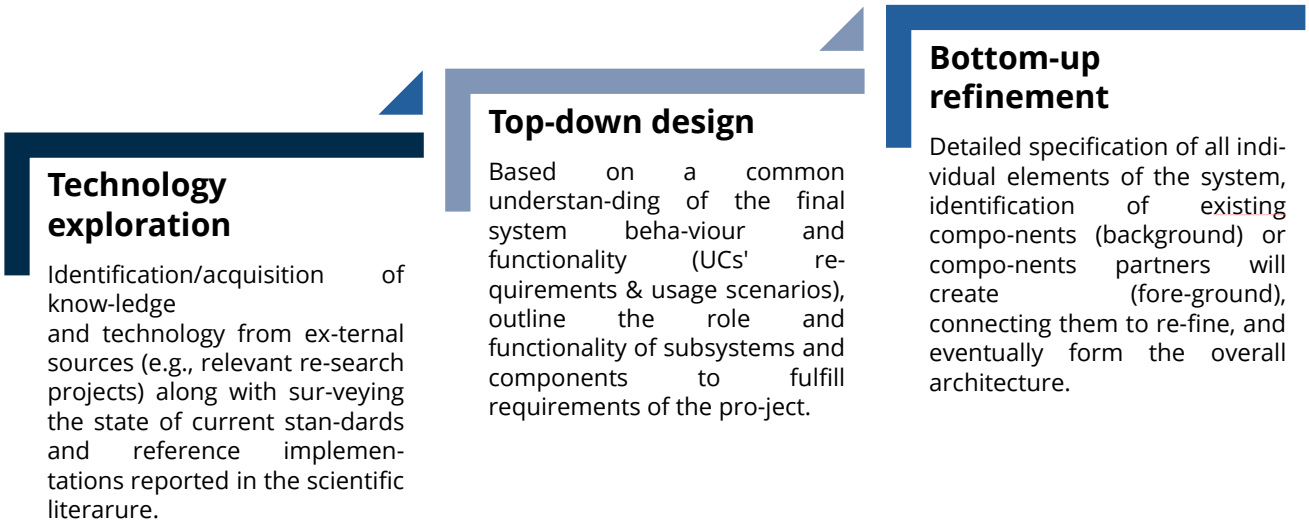


Figure 4: Architectural design approach within OPTIMAI (retrieved from D2.4, M12).

3.1 Technology exploration

Technology exploration encompasses a procedure where technologies and architectural models relevant to the OPTIMAI goals and vision are identified and studied, in order to generate inputs from relevant results reported in the frameworks of other national and international (preferably, EU-funded) Research & Development (R&D) activities, and distil them into the OPTIMAI architecture design.

Section 3, D2.4, M12.

Technology exploration with respect to the architectural update refers to the further elaboration of identified technologies to be brought into the architecture (i.e., partners' background). This has been applied by thoroughly studying the various deliverables of the project, in which these technologies have been elaborately described. This has enabled the OPTIMAI system architects to identify architectural requirements related to information exchange between modules (such as the definition and refinement of exposed and consumed APIs to and from architectural components), as well as proposed deployment considerations and characteristics that relate to the system's topological description.

The list of deliverables surveyed is as follows:

Table 2: Deliverables surveyed as part of the second technology exploration architectural design phase

Deliv. Number	Deliv. Title	Addressed Modules/Subsystems (D2.4)
D3.1	Multisensorial data acquisition and actuation network	Quality Control Sensors Network Middleware On-the-edge processing for acquisition and actuation (OPR4AA)
D3.8	Blockchain framework for traceability and data integrity - 1st version	Blockchain Framework components
D3.10	AI Components for Quality Control toward Zero-Defect Manufacturing	AI-based Production Monitoring Component
D6.1	Decision support and early notification framework_1st version	OMIDES Back-end OMIDES Front-end Manufacturing (re-)configuration Service Defect Detection and Quality Control Service
D6.3	Intelligent Marketplace for AI Sharing and Scrap Reuse - 1st Version	Intelligent Marketplace Back-end OPTIMAI Intelligent Marketplace Dashboard

3.2 Top-down design

Top-down design is the process of dividing the system into the elements it is composed of, leading to the identification of any-level subsystems and base components. The purpose of this phase is to gain an understanding of the main function of the components and how they will communicate with each other.

Section 3.1, D2.4, M12.

Following the first version specification of the OPTIMAI Architecture and technology exploration, the top-down design of the architecture was refined, so as to include all newly introduced components (e.g., the OPTIMAI Agent at the QCS Network), as well as merge, refine, break-down or remove obsolete components (i.e., individual sensors) from the D2.4 version. To this end, the vertical segmentation of the architecture as presented in D2.4 was maintained, and components were placed inside the larger subsystem where they were deemed more appropriate based on their deliverable descriptions. Initial flows to and from components were re-designed, so as to be abstractly described and prepared for presentation to the project partners in the ensuing

bottom-up phase. In cases where such descriptions were not clear, or were ambiguous, individual partners were contacted to elaborate further. In some cases, components were further broken down, or placed within different subsystems altogether.

Eventually, the top-down view of the architecture yielded **38** total components and modules, which are presented in Table 3, below. An asterisk (*) is used to indicate that a component was identified during the refined technology exploration phase, and is therefore not mentioned in D2.4, while a revision superscript ([r]) is used to indicate a component with a revised name and role (with the previous corresponding component, if applicable, in brackets).

Table 3: List of identified main and total functional blocks in the OPTIMAI top-down architectural stack

Node	Component No.	Component(s)	Partner(s) responsible	Implementation Example (Product, or document Section)
Quality Control Sensors Network	001	OPTIMAI Agent*	FINT	Section 14.1.1.1
	002	OPTIMAI SoftSensor*	CERTH UNIMET	Section 14.1.1.2
	003	EyeVision Web Service*	EVT	Section 14.1.1.3
Edge Computing Modules[r]	-	Middleware	FINT	Section 4.1.3
	-	On-the-edge processing for acquisition and actuation (OPR4AA)[r] (AI Edge Processing Service module)	ENG	Section 4.1.4
Middleware	004	Cybersecurity Defence Module	FINT	Section 14.1.1.4
	005	IoT Agent[r] (Multimodal Data Collection Agent)	FINT	Section 14.1.1.5
	006	FINoT Platform	FINT	Section 14.1.1.6
	007	Middleware Service	FINT	Section 4.1.3.4
OPR4AA[r]	008	Data Flow Controller*	ENG	Section 14.1.1.8

	009	Analytics Engine*	ENG	Section 14.1.1.9
	010	Hadoop Distributed File System (HDFS)*	ENG	Section 14.1.1.10
Cloud Computing Modules[r]	-	Middleware Cloud Data Repository	FINT	Section 4.1.6
	-	Blockchain Framework[r]	CERTH	Section 4.1.7
	-	Operator-Machine Interaction & Decision Support (OMIDES) Back-End[r]	CERTH	Section 4.1.8
	-	Intelligent Marketplace Back-End[r]	FINT	Section 4.1.9
	-	AI Framework[r]	UTH CERTH	Section 4.1.10
	-	DT Framework[r]	VIS	Section 4.1.12
Middleware Cloud Data Repository	011	File Storage	FINT	Section 14.1.1.11
	012	Historical Data	FINT	Section 14.1.1.12
	013	Open Datasets	FINT	Section 14.1.1.13
Blockchain Framework[r]	014	Blockchain API Service*	CERTH	Section 14.1.1.14
	015	IPFS Node	CERTH	Section 14.1.1.15
	016	Access Control Smart Contract[r] (Access Control Service)	CERTH	Section 14.1.1.16
	017	Firmware/Software Validation Smart Contract[r] (Firmware validation Service)	CERTH	Section 14.1.1.17
	018	Model and Data Integirty Smart	CERTH	Section 14.1.1.18

		Conteact[r] (Data Integrity Service / AI Model Integrity Verification Service)		
OMIDES Back-End[r]	019	Pose Estimation Service	CERTH	Section 14.1.1.19
	020	Activity Recognition Service	CERTH	Section 14.1.1.20
	021	Instance Segmentation Service	CERTH	Section 14.1.1.21
	022	Decision Support System (DSS) Engine*	CERTH	Section 14.1.1.22
	023	Conversational Agent Back-end*	CERTH	Section 14.1.1.23
Intelligent Marketplace Back-End[r]	024	Marketplace Back-end Service*	FINT	Section 14.1.1.24
	025	Identity Management System (IDM)[r] (Marketplace Authorisation)	FINT	Section 14.1.1.25
	026	AI Algorithm Catalogue*	FINT	Section 14.1.1.26
	027	Scrap Reuse Catalogue*	FINT	Section 14.1.1.27
AI Framework[r]	028	Manufacturing (re-) configuration Service	CERTH FORTH	Section 14.1.1.28
	029	AI-based Production Monitoring Component[r] (Production Monitoring & Quality Control Service)	UTH	Section 14.1.1.29
	030	Defect Detection and Quality Control	CERTH	Section 14.1.1.30

		Service[r] (Defect Detection & Prediction Service)		
DT Framework[r]	031	DT Process Models[r] (Process Digital Twins)	VIS	Section 14.1.1.31
	032	Virtualized Sensors Network	VIS	Section 14.1.1.32
End-users' applications[r]	033	OPTIMAI Intelligent Marketplace Dashboard[r] (OPTIMAI Intelligent Marketplace Customer Front-end)	FINT	Section 14.1.1.33
	034	Visual Components Software[r] (Visual Simulation Engine)	VIS	Section 14.1.1.34
	038	MAIAR Software*	FORTH	Section 14.1.1.38
	-	OMIDES Front-end[r]	CERTH	Section 4.1.14
OMIDES Front-end[r]	035	Early Notification Framework*	CERTH	Section 14.1.1.35
	036	Visual Analytics[r]	CERTH	Section 14.1.1.36
	037	Conversational Agent Interface*	CERTH	Section 14.1.1.37

3.3 Bottom-up refinement

Bottom-up design refers to the process in which the components of a system are specified in detail. During this process units are linked to form a more complex system. For the specification of the OPTIMAI architecture, the bottom-up approach was subsequent to the top-down approach. During this phase, technical partners provided their input with regard to the technologies and software components they are planning to contribute to the project. These may be components that partners bring to the project as part of their background, or they have developed as part of the work carried out in the framework of the project Tasks (foreground).

Section 3.2, D2.4, M12.

Architecture refinement for the period between M12 (D2.4) and M18 (D2.5) concluded with a second bottom-up refinement phase, in which all collected components listed in Table 3, as well as those to be reported in upcoming deliverables (identified per Task), were elaborated and

given substance through a final, elaborate component refinement template (see Appendix A). The exercise aimed at offering insight into the architectural components required for the technical realisation of the information flows specified for the project use case (Section 6). The template served as a means of gathering information, especially about the novel functional blocks identified in the top-down specification (Section 3.2), which eventually form the concrete OPTIMAI architecture.

The specification was undertaken through two architectural workshops held in Weeks 21and 23 of 2022, which aimed at presenting the refinements based on the outputs of technology exploration and top-down specification (Week 21) as well as gather final needed information and insight on components, whose templates were gathered (Week 23). Along with the role and responsibilities of the components, additional information was gathered, such as COMPONENT DIAGRAMS, exposed interfaces, dependencies on software (and hardware) elements, and a deployment diagram to form a basis of the deployment architectural view. The outcomes of this exercise constitute the contents reported in Section 4 of this deliverable.

3.4 Ethical compliance – Compliance through Design

Void (the contents provided in Section 3.3 of D2.4 apply).

4 OPTIMAI Architecture - Functional view

This Section presents the final list of architectural elements and building blocks (both individual components and subsystems, as identified in Table 3) that together combine to deliver the complete functionality of the OPTIMAI system. The following paragraphs will describe these components roles and responsibilities within the overall function of the OPTIMAI framework, along with the synergies that they should implement with other components, both internal and external to the OPTIMAI system. Each component and subsystem will be further elaborated by means of a Unified Modelling Language (UML) component diagram, providing a view on the static implementation of each system and subsystem. The high-level generic, service-oriented architectural diagram of the final system is depicted in Figure 5.

> *This architectural view presented follows the approach of segmenting the envisioned IT systems on a vertical axis, thus adopting a layered design that indicates the high-level classification of the different technological components in accordance to their contribution to the key user requirements of the project use cases. The functional architecture stack depicted in Figure 5 is further aimed at highlighting rudimentary principles of the OPTIMAI development over time, such as the properties, relationships and execution environment of the functional elements. Each layer corresponds to a major subsystem driving the flow of information from top to bottom (i.e., from the sensing IoT hardware all the way to the users' equipment). These subsystems/nodes are: (i) the Quality Control Sensors Network; (ii) the Edge Computing Modules; (iii) the Cloud Computing Modules; and (iv) the Users' Applications. Specific alignment to the deployment characteristics of each subsystem are discussed in more detail in Section 6. The purpose of this Section is to instead highlight the analogy of use case requirements to the different functional blocks' responsibilities (i.e., making sure component functionalities are well-represented, and that they can be properly matched against elicited system requirements). The descriptions are provided at a functional level, which omits confronting specific implementation details at architectural level, as they are particular to the development Task delivering each component to the final integrated system.*
>
> *In addition to the definition of components' behaviours and interrelations, care has been taken to refine the original architecture reference model (discussed in Section 2.3) in accordance to well-defined standards (as discussed in Section 2) so as to support interoperability. Such an alignment enables extension of the OPTIMAI solution in a manner that would enable its replication for use cases outside the ones explicitly defined in the scope of the project, within the context of both I4.0 (via a direct mapping of OPTIMAI to the RAMI 4.0 cubic model) and IIoT (by applying mappings drawn between IIRA and RAMI 4.0) scenarios.*
>
> Section 4, D2.4, M12.

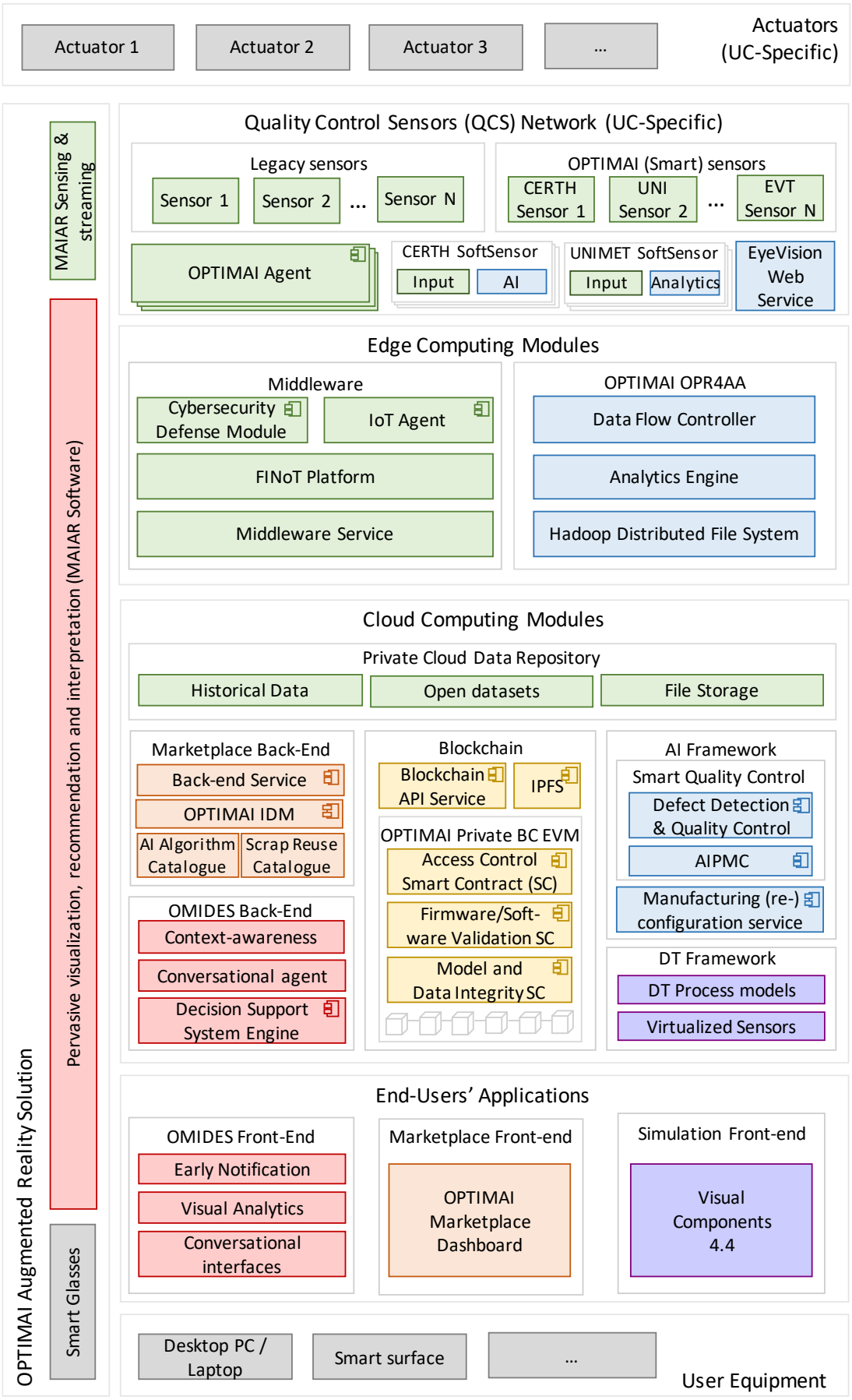


Figure 5: OPTIMAI Functional architecture view – component diagram.

In addition to the vertical segmentation, the architectural stack presented in Figure 5 is further distinct into key emerging technologies being adopted into I4.0 enterprises to realize "smarter" manufacturing processes and less wasteful production. These are:

- Multi-sensory data acquisition (green);
- AI analysis (blue);
- Distributed ledger technologies (yellow);
- Context-aware recommendation (red); and
- Digital-twinning for production optimization inference (purple).

*The remainder of this Section is organised as follows: Section 4.1 delivers on the descriptions for the subsystems and components depicted in **Figure 5**, which are directly derived from the templates handed out to project technical partners during the bottom-up design specification, as discussed in Section 3.3. With the functional perspective in place, Section 4.2 then discusses the alignment of the OPTIMAI architecture with the two major standards-led reference architectures, RAMI 4.0 and IIRA.*

Section 4, D2.4, M12.

4.1 OPTIMAI functional blocks

*The following sub-sections provide a description for each of the functional blocks (both components and larger subsystems) comprising the OPTIMAI system as illustrated in **Figure 5**. Wherever possible, core system entities (at any-level, i.e., component or subsystem) will be described also in relation to its interdependencies with other OPTIMAI components (thus elaborating on its role in the overall information structure supported by the architecture), as well as the primary interactions formed with other functional elements by means of expected inputs and outputs driving the foreseen runtime behaviour*

Section 4.1, D2.4, M12.

The subsystem and descriptions below are derived from both the technology exploration phase as well as the high-level documentation of the various functional aspect of each component delivered by that component's owner during the bottom-up specification stage of the architecture (re-) definition. The latter was carried out through the elaborate component refinement templates (pages 3-6) included in this report in Appendix A. Where relevant, high-level component diagrams are used to better illustrate structure and interdependencies among components.

4.1.1 Quality Control Sensors Network

The Quality Control Sensors (QCS) Network subsystem is comprised of all IoT sensor devices employed for data collection regarding current production parameters. A variety of device types have been identified during the first year of the project in the context of Task 3.1 "Multisensorial data acquisition and actuation network", and a complete list of devices and specifications will be provisioned with D3.1 due in M16.

Section 4.1.1, D2.4, M12.

The original elaboration of the QCS Network closely mirrored the provisions of the conceptual architecture established at project preparation level (before signing of the GA), and thus focused on sensors at a hardware-level. Since the release of D3.1, the QCS Network has been elaborated to include a distinct definition for hardware and software, or softwarized components. Hardware-wise, the QCS Network consists of the range of **existing (legacy) sensors** in each of the pilot sites (specified in Section 3.1 of D3.1), alongside those developed within the lifetime of the OPTIMAI, specifically for the project use cases (as elaborated in Section 3.2 of D3.1). These latter sensors will be developed by OPTIMAI Consortium members CERTH, UNIMET and EVT, and all involve higher level functions executed either stand-alone, or utilising direct communication with a nearby PC (e.g., Power over Ethernet - PoE). These sensors will be extended by software components that can execute industrial vision applications, hence giving substance to the **OPTIMAI "smart" sensors** array.

To enable the intercommunication of the hardware components with the OPTIMAI software platform, additional modules were introduced to the internal QCS Network architecture, as depicted in Figure 5. The remaining paragraphs in this Section will aim at describing these elements and rationalize their addition to the OPTIMAI architectural stack.

4.1.1.1 001 OPTIMAI Agent(s)

The template for the **OPTIMAI Agent** component was originally described in Section 2.4 of D3.1. In that document, OPTIMAI Agents were explicitly defined as *"a software program that acts as a network router, routing sensor data between the sensors and the Middleware. They can handle both inbound and outbound traffic. Outbound traffic streams are used for sending sensor data to the Middleware, while inbound traffic is used for receiving action commands".* At least three variants to this OPTIMAI Agent template will be developed, one for each of the pilot sites, to accommodate the intricacies of the pilot sites' data collection platforms, e.g., the pre-existing sensors and IoT devices installed, communications protocols used, manufacturing data APIs used for obtaining access to the collected data, etc. For a complete overview, readers are referred to D3.1.

The purpose of the OPTIMAI Agent inside the architecture is to provide a means for the system to obtain data from pilot sites' platforms and databases (where *existing sensors* deposit data – a process that is not subject to change for the purposes of the OPTIMAI project) and forward it to the *Middleware* (which does not poll for data in and of itself). In addition, and similar to the data acquisition pipeline, the OPTIMAI Agent is used to forward and apply actuation commands coming from the Middleware to the actuators inside the OPTIMAI pilot projects (as described in Section 4 of D3.1).

Key functional aspects and features provided by the OPTIMAI Agent(s) include the following:

- Collect and filter existing sensors' data by implementing the the proper communication protocols with the pilot's existing data collection platform.
- If necessary, translate sensor data to the Middleware's communication protocols.
- Send the data to the Middleware using one of the available interfaces the latter exposes.
- Subscribe on specific topics for actuation commands.

The high-level functional architecture of the OPTIMAI Agent is depicted in Figure 6:

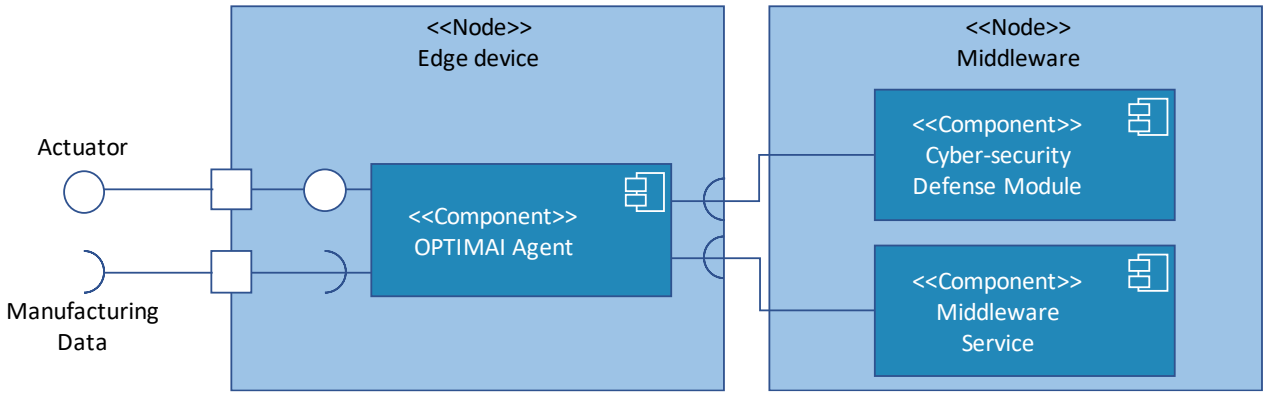


Figure 6. OPTIMAI Agent component diagram.

4.1.1.2 002 OPTIMAI SoftSensor(s)

A Software Sensor (SoftSensor) is a software that integrates and processes a number of measurements coming from one or more hardware sensors, and uses these measurements to generate new, higher-order measurements by executing a range of different algorithms, including, AI-based segmentation, object detection and image classification. The purpose of the **OPTIMAI SoftSensors** is to essentially "create" additional measurements utilizing industrial vision applications, i.e., as if a dedicated hardware sensor was deployed to obtain those measurements.

Within the context of OPTIMAI, at least two variants of the SoftSensor template are defined (as specified in D3.1):

- The ***CERTH SoftSensor***, which comprises a 2D area scan camera and an NVIDIA Jetson module that is used to run deep convolutional network algorithm to regress 3D quality inspection metrics, such as the volume of glue deposited on PCBs in the MTCL pilot site.
- The ***UNIMET SoftSensor***, which combines the proprietary UNIMET Optiscan scanner and UNIMET M3 data analytics software as a means to propagate post-processed data (i.e., dimensional assessment of point cloud data) to the Middleware.

The high-level functional architecture of the OPTIMAI SoftSensors is depicted in Figure 7:

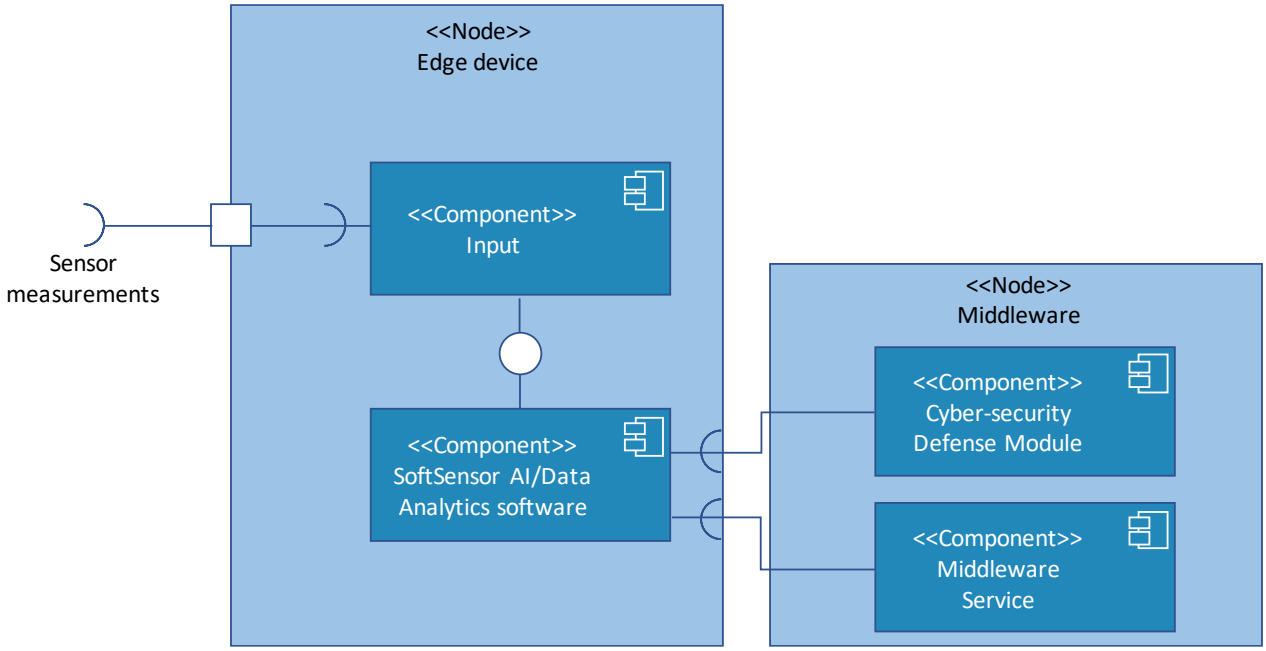


Figure 7. SoftSensors component diagram.

4.1.1.3 003 EyeVision Web Service

The **EyeVision Web Service** is a core functional block provisioned by EVT. It is a core component of EVT's industrial vision sensors, offering AI-processing and seamless process integration with the OPTIMAI architecture. The system is capable of executing different industrial vision applications. It supports the acquisition of 1D, 2D and 3D images, including input from thermal or multi spectral cameras, and offers a range of different algorithms to be executed, including AI-based segmentation, object detection and image classification. In addition, it implements communication clients based on various protocols (e.g., OPC UA, MQTT, REST, alongside other state-of-the art protocols) to establish communication with the Middleware.

Key functional aspects and features provided by the EyeVision Web Service include the following:

- Support for single, or multi camera systems, and capability to combine different image types.
- A user of the software can configure the application based on an extensive tool box of algorithms.
- Support for deep learning based neural networks provisioned through built-in Graphical Processing Unit (GUI) used for annotation and training (on specific platforms).
- Scalable systems (hardware independent).
- Communication interfaces support a big range of protocols (MQTT, OPC UA, Profinet, Modbus, etc).

An important capacity supported by the EyeVision Web Service, as per the provisions of Section 5.4.2 of D3.1 is the capacity of the module to provide a C++ plugin interface allowing for the deployment of lightweight enough algorithms to be deployed and executed directly into the EyeVision software, hence bypassing the Middleware. This is especially useful for time-critical

operations (e.g., minor calibrations that should be executed within milliseconds), rapid execution and reducing of downstream computations is highly desirable.

The high-level functional architecture of the EyeVision-enabled EVT industrial sensors is depicted in Figure 8:

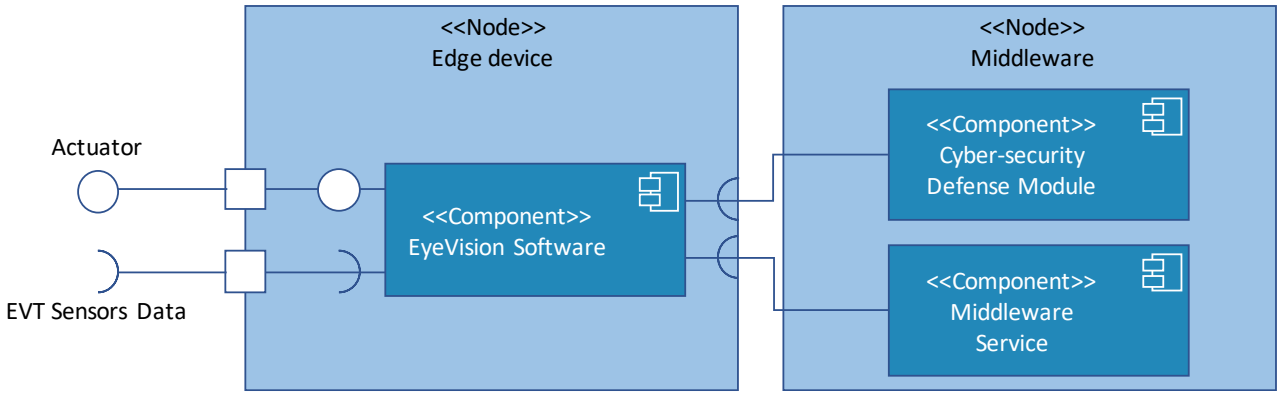


Figure 8. EVT Industrial vision sensors with EyeVision Web Service component diagram.

4.1.2 Edge Computing Modules

Edge computing is emerging as a critical enabler in smart manufacturing, enabling the rapid execution of code originally intended for Cloud computing resources on either the devices themselves, or a gateway device or Personal Computer (PC) in close proximity. To capitalise on this emerging paradigm, the OPTIMAI architecture incorporates subsystems and modules for deploying on-the-edge intelligence regarding monitoring and control of specific ("smart") sensors in the QCS Network subsystem [...]. This enables OPTIMAI to dynamically manage the various acquisition parameters.

Section 4.1.2, D2.4, M12.

In support of this edge node architecture, the OPTIMAI Edge Computing Modules incorporate the following subsystem modules:

- *the **Middleware subsystem** (Section 4.1.3), which provisions services for: (i) the collection of all sensors' data in real time; (ii) the application of cybersecurity at the acquisition level (as soon as data enters the system); (iii) sensor health monitoring functions; and (iv) coordination of the exchange of information between the edge and cloud modules.*

Section 4.1.2, D2.4, M12.

- the **On-the-edge processing for acquisition and actuation (OPR4AA) platform** (Section 4.1.4), which enables the deployment and operation of AI services on-the-edge, as specified in D3.1.

4.1.3 Middleware Subsystem

The Middleware subsystem provisions secure interfaces between the various units, components, sensors and subsystems. It supports the network protocols required to exchange control and store data and information needed to facilitate operations and services in an

environment with many different networking and system components. As such, the Middleware subsystem incorporates all necessary services required for integrating the various sensor devices in the QCS Network within a unified framework. Aside from data collection, a core purpose of this subsystem is to expose a unified interface to interact with the other OPTIMAI platform components, exposing all necessary functionalities by establishing all necessary API endpoints (based on the Representational State Transfer - REST architecture) towards the interdependent platform components.

This subsystem is based on FINT's commercial FINoT Platform solution, albeit extended to implement the necessary functionality dictated by the project objectives. Based on the architecture of the FINoT solution, the Middleware subsystem will provision functions toward: (i) sensors' data acquisition and registration; (ii) facilitating storage, processing and aggregation (on the edge); (iii) incorporate cyber-security defence in the form of authentication/authorisation routines; and (iv) expose functions through a powerful REST API interface.

Section 4.1.3, D2.4, M12.

The Middleware provides the necessary interfaces for communication with devices, sensors and actuators. It is responsible for data collection, management and querying; provides a unified storage layer for object, files and historical data; provides Complex Event Processing (CEP) and exposes external module interfaces for other subsystems within the OPTIMAI architecture.

The high level functional architecture of the Middleware is depicted in Figure 9.

4.1.1.4 004 Cybersecurity Defence module

Void (the contents provided in Section 4.1.3.4 of D2.4 apply).

4.1.1.5 005 IoT Agent

Void (formerly known as the *Multimodal Data Collection Agent*, the contents provided in Section 4.1.3.1 of D2.4 apply).

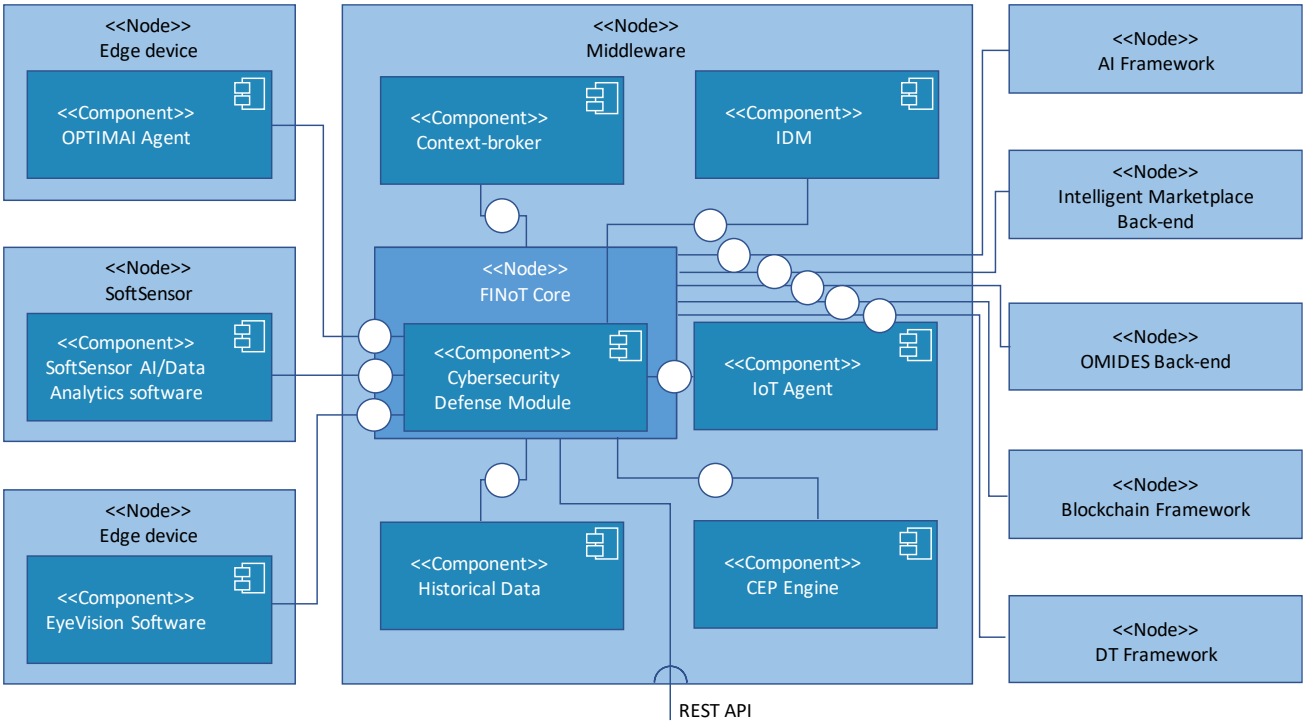


Figure 9. Middleware component diagram

4.1.1.6 006 FINoT Platform

Void (the contents provided in Section 4.1.3.2 of D2.4 apply).

4.1.1.7 007 Middleware Service

Void (the contents provided in Section 4.1.3.3 of D2.4 apply).

4.1.4 On-the-edge processing for acquisition and actuation (OPR4A)

The **OPR4AA Platform** is an extension of ENG's Digital Industry Data Analytics (DIDA) Platform, which was originally described in Section 5 of D3.1. In that document, the was explicitly defined as *"a FIWARE/Apache based platform for Smart Industry that supports a variety of connectors / components for every aspect of Smart Data Management and Integration as well as Data Persistence, Data Sovereignty and Data Security"*. Hence, the OPR4AA directly relates to the AI services developed in OPTIMAI, and targets sensor acquisition optimization; real-time analytics modules; and adaptation of intelligent manufacturing assets.

Intricate details about the implementation of the OPR4AA Platform are provisioned in D3.6, delivered in parallel to the present report. For architectural purposes, a brief overview of the key features and internal components will be elaborated in the following paragraphs.

Key functional aspects and features provided by the OPR4AA Platform include the following:

- It enables pre-processing of sensors data for an efficient dispatching to other OPTIMAI components, and deployment of AI services on-the-edge, reducing execution latency.
- It allows analysis of big data using AI algorithms running on top of the DIDA modules (briefly described in the following paragraphs).

- It enables data acquisition, pre-processing of data, and the overall enhancing of the on-the-edge "smartness" of sensors.

The high-level functional architecture of the OPR4AA Platform is depicted in Figure 10:

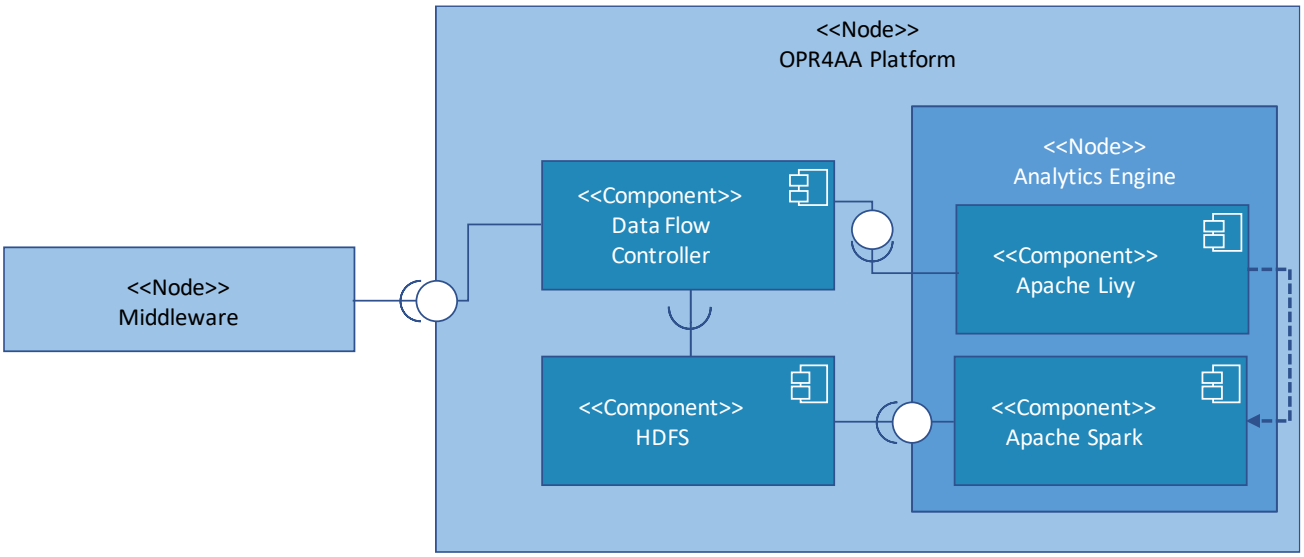


Figure 10. OPTIMAI OPR4AA Platform component diagram.

4.1.1.8 008 Data flow controller

The OPR4AA Platform **Data flow controller** is a component responsible for managing, distributing and automating flow of data between the Middleware and the other sub-modules of the Platform. It is based on FIWARE Draco, a Generic Enabler based on the Apache NiFi dataflow system. As such, it is responsible for exposing the FIWARE Draco interface consumed by the Middleware. In addition, it can also invoke HTTP calls to the Middleware API for distributing data to it, thus implementing a two-way information flow between the two functional blocks (see Figure 10).

4.1.1.9 009 Analytics engine

The OPR4AA Platform **Analytics engine** refers to the modules responsible for the execution of AI Algorithms on the edge. As was predicated in D3.1, it will be based on the Apache Spark[1] engine, executing (mostly) Python-based AI algorithms, without however excluding the capacity to bridge the execution of algorithms based on different architectures, pipelines and languages. To invoke job execution tasks for the AI algorithms on Spark, the Apache Livy[2] REST Service will be employed, receiving input data from the Data flow controller. AI results will be returned to the latter component so as to be communicated back to the Middleware.

4.1.1.10 010 Hadoop Distributed File System (HDFS)

OPR4AA Data storage will implement the **Hadoop Distributed File System (HDFS)**, intended for the reliable storage of very large files (e.g., such as the ones produced by EVT point cloud data) over distributed physical resources. The Analytics engine (Spark) will be configured to both read

[1] https://spark.apache.org/
[2] https://livy.apache.org/

and write data to and from HDFS, since the two can be deployed together utilising various deployment options.

4.1.5 Cloud Computing Modules Subsystem

The Cloud Computing Modules subsystem contains the OPTIMAI components that are expected to be deployed in a cloud computing environment, due to their foreseen needs in storage and computational power (e.g., for processing-heavy AI and other routines).

Section 4.1.4, D2.4, M12.

The OPTIMAI Cloud Computing Modules consist of the following subsystems and modules:

- *the **Middleware Cloud Data Repository subsystem** (Section 4.1.6), which acts as the centralised storage point of the entire OPTIMAI system.*
- *the **Blockchain subsystem** (Section 4.1.7), which is responsible for maintaining a distributed ledger of all critical operations, the employment of smart contracts to automate several production processes, and the provision of data integrity verification mechanisms.*

Section 4.1.4, D2.4, M12.

- the **Operator-Machine Interaction & Decision Support (OMIDES) Backend subsystem** (Section 4.1.8), which utilizes the results of the defect detection and prediction, status of the production line and operation (both originating at the AI Framework), and simulation routines (at the DT Framework), provides the information needed to perform re-configuration automations.

- *the **Intelligent Marketplace Back-End subsystem** (Section 4.1.9), which will facilitate user-generated data storage and transactions functionality for the envisioned OPTIMAI scrap and AI models marketplace solution.*

Section 4.1.4, D2.4, M12.

- the **AI Framework** (Section 4.1.10), which executes smart quality control processes and calculates production optimization parameters based on predictive analytics using the powerful resources available the cloud.
- The **DT Framework** (Section 4.1.11), which provides the resources necessary for the virtualization and simulation of the production line.

4.1.6 Middleware Cloud Data Repository Subsystem

Void (the contents provided in Section 4.1.5 of D2.4 apply).

4.1.1.11 011 File Storage

Void (the contents provided in Section 4.1.5.1 of D2.4 apply).

4.1.1.12 012 Historical Data

Void (the contents provided in Section 4.1.5.2 of D2.4 apply).

4.1.1.13 013 Open Datasets

Void (the contents provided in Section 4.1.5.3 of D2.4 apply).

4.1.7 Blockchain Framework

The Blockchain subsystem will be responsible for dealing with system objectives regarding the security, privacy, traceability, integrity, compatibility and interoperability of data storage and exchange. In addition, a blockchain-based AI model integrity validation mechanism will be provisioned. The foreseen developed Cloud-based subsystem will be provisioned by implementing a Blockchain-as-a-Service (BaaS) model. In this regard, a blockchain Application Programming Interface (API) will be exposed to enable traceability and validity of every data transaction occurring in the system, whether it be a collection of measurements, a reconfiguration request, or a sensor health check.

Section 4.1.6, D2.4, M12.

The Blockchain framework was described in full (including implementation details) in D3.8. Its responsibilities include firmware validation, access control, and data integrity, as well as developing a logging/auditing method for the system's important functions, such as sensor actuations. All of the important system operations are recorded as immutable and verifiable transactions in a blockchain network in the context of this data exchange. In accordance to Section 2.6 of D3.8, it is implemented in a Private Ethereum network, using a "Proof of Authority" consensus mechanism.

Smart contracts, i.e., programs that can be executed inside an Ethereum Virtual Machine (EVM), are used to automate a range of tasks inside the manufacturing line. Therefore, the specific objectives pursued by the Blockchain within the OPTIMAI architectural stack are organised in the following Smart Contract (SC) modules:

- **Access control** (Section 4.1.1.16): An access control mechanism is integrated to prevent unauthorized users from carrying out important system operations.
- **Firmware and software validation** (Section 4.1.1.17): All important system processes will be recorded as immutable and verifiable transactions.
- **Model and Data Integrity** (Section 4.1.1.18): The integrity of the software and firmware versions installed, the AI models used, and the measurements supplied by the system's sensors using blockchain-based data integrity methodologies.

The high-level functional architecture of the Blockchain Framework is depicted in Figure 11.

Individual components depicted in this diagram will be presented in more detail in the paragraphs below.

4.1.1.14 014 Blockchain API Service

The **Blockchain API Service** is described in D3.8. It will be a gateway component exposing a northbound interface toward the Middleware accepting HTTP POST requests to receive data from the QCS Network. It further exposes a southbound interface to communicate with the

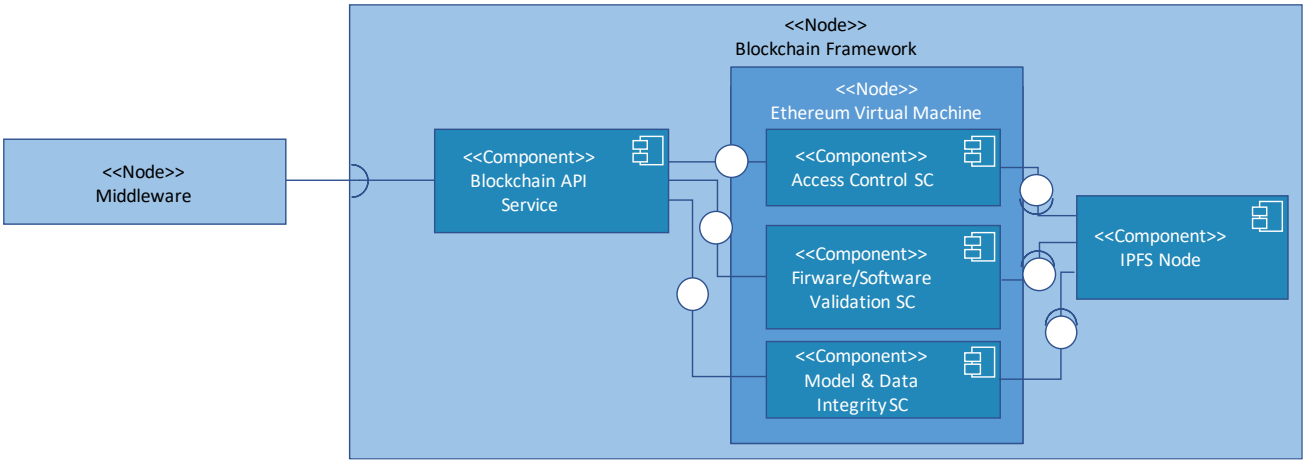


Figure 11. OPTIMAI Blockchain Framework component diagram

Blockchain smart contracts with the use of Web3.js endpoints. In addition, it will integrate with an InterPlanetary File System (IPFS) node (see Section 4.1.1.15) to obtain unique hash values for each data stored on the Ethereum blockchain/side chain architecture elaborated in D3.8.

4.1.1.15 015 IPFS Node(s)

IPFS Nodes are used to implement the content-addressed block storage system needed for the Blockchain framework. Its role, in accordance to D3.8, is to calculate a unique hash value for data and provide it to the smart contracts, which will then store this value as transactions over Ethereum. IPFS Nodes hence implement both distributed data storage, and encryption. Depending on the amount and scale of data, one or more private IPFS Nodes may be needed per pilot site.

4.1.1.16 016 Access Control Smart Contract

The **Access Control Smart Contract** has been elaborated in D3.8 as implementing the *"functionality of the system for allocating or denying authorization to a user based on his request to perform an action on a protected resource or object"*. It is a smart contract EVM program that operates on top of the blockchain, used to validate and execute Role-Based Access Control (RBAC) mechanisms to determine whether a user requesting data from the Blockchain has the proper rights to access the hash of the requested data. This is achieved by exposing a REST API toward the Middleware, which, as specified in D3.8, will "*contain information about the Ethereum address, the role of the address, the Authorization of the role and the permission of the role. This information will be stored on the private Ethereum network*".

4.1.1.17 017 Firmware/Software Validation Smart Contract

As specified in D3.8, the **Firmware/Software Validation Smart Contract** is an EVM program used to verify transactions related to firmware updates and the assigning of a software configuration to a sensor (i.e., one with built-in configuration agent, such as EVT sensors). Its role is to assert validity of said firmware or software configuration, by storing a new block to the distributed ledger that will include information, such as the IP address of the sensor, the versions of the firmware/software prior to and after the update, a timestamp and a hash value for the new firmware/software obtained by the IPFS. This hash is then used by the sensor agent in a

subsequent query to the blockchain (forwarded whenever a binary is sent to the senor for updating) to assure validity of the update process.

4.1.1.18 018 Model and Data Integrity Smart Contract

The **Model and Data Integrity Smart Contract**, as explained in D3.8, is an EVM program used to *"enable the immutable record of AI system choices and activities, resulting in a more trustworthy AI"*. The contract will be utilised to keep track of both the AI model, as well as the data used to develop this AI model. This will allow traceability of the AI model evolution, since it allows user to check not only the model itself, but also the learning history behind it. This verification mechanism is employed so that operators can be assured of using the correct AI model for a given operation, by comparing hash values of the model to be deployed with the one intended.

4.1.8 OMIDES Back-End Subsystem

The **OMIDES Back-End** subsystem has been described in Section 2.1 of D6.1 as providing *"the context in which the operator is working at a given time (e.g. user's task, post), the operator's relative position with equipment (to know which is the right person for imminent (re)-configuration), as well as the defective products detected in relation to the operators' position (to act accordingly)"*. In this respect, the key features of the OMIDES Back-End are summarised as follows:

- Provide the OPTIMAI stack with **context-awareness** components toward realizing an *early notification framework*, by employing:
    - **Activity Recognition:** detect and recognize the activities of the operator in the shop floor (e.g., tasks, gestures)
    - **Instance Segmentation:** detect objects of interest and separate them from the background by producing pixel-wise segmentation masks. Objects of interest are considered to be the produced parts and their constituent modules and production machines.
    - **Pose Estimation:** estimate the pose of the objects of interest (produced parts, production machines) in the AR environment, thus obtaining their relative position with respect to the operator.
- Facilitates the storage of user profiles and generation of alerts targeted at the appropriate individual on the shop floor (by means of a Decision Support System [DSS] engine, see Section 4.1.1.22), to perform re-configurations manually on the appropriate device (running the OMIDES Front-end), and allowing for fast response depending on the current context.
- Implements the necessary cloud-computing modules for enabling voice-enabled natural language interactions with a conversational virtual agent (Section 4.1.1.23).

The high-level functional architecture of the entire Decision support system and early notification framework, as described in D6.1 (including the OMIDES Front-end components) is depicted in Figure 12, below:

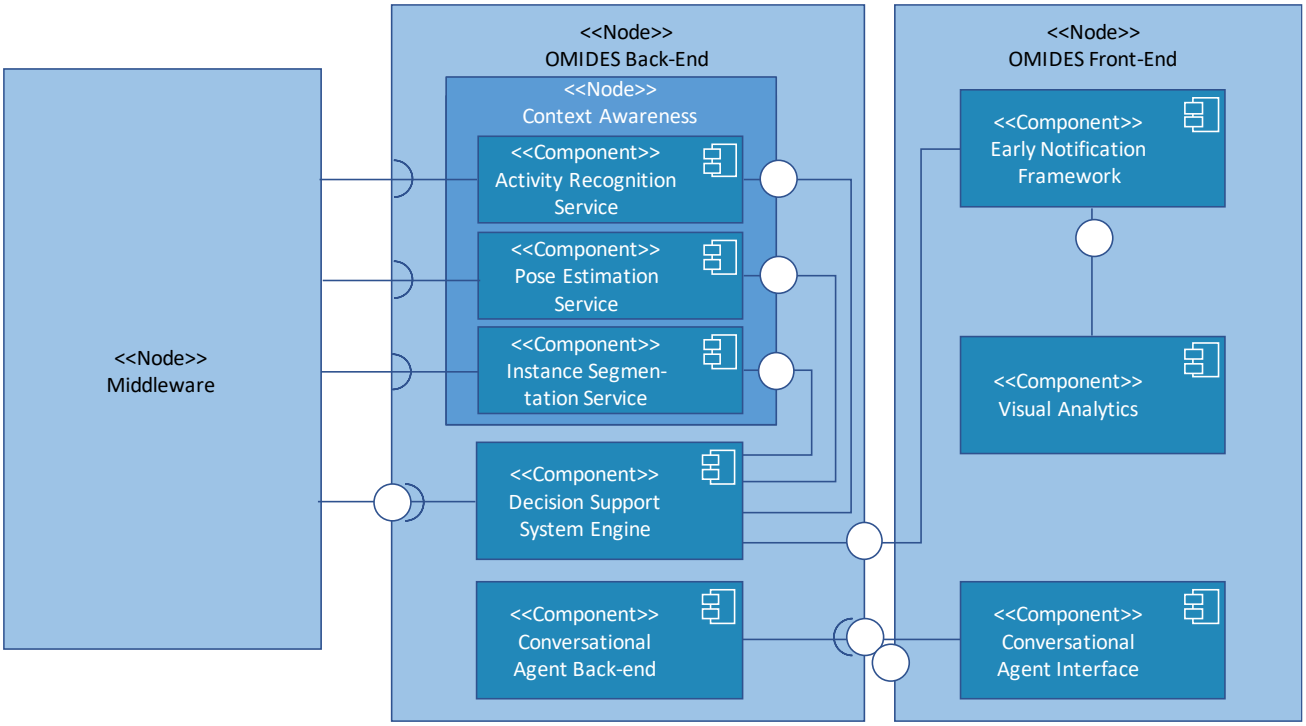


Figure 12. OPTIMAI Decision support system and early notification component diagram

Individual components depicted in this diagram will be presented in more detail in the paragraphs below, as well as in Section 4.1.14.

4.1.1.19 019 Pose Estimation Service

Void (the contents provided in Section 4.1.7.1 of D2.4 apply).

4.1.1.20 020 Activity Recognition Service

Void (the contents provided in Section 4.1.7.2 of D2.4 apply).

4.1.1.21 021 Instance Segmentation Service

Void (the contents provided in Section 4.1.7.3 of D2.4 apply).

4.1.1.22 022 DSS Engine

The **DSS Engine** will be responsible for storing and maintaining user profiles (constructed through the use of a conversational agent, as explained in the following paragraph), which will aim at supporting adaptable user interfaces at the OMIDES Front-end level (thus tailoring the front-end experience in accordance to the collected user preferences and employee profile), as well as ensure that generated data, quality results and potential alerts communicated to the OMIDES Back-End originating at the AI Framework level, are forwarded to the appropriate control terminal and the user authorized to view and take action.

4.1.1.23 023 Conversational Agent Back-End

In accordance to Section 5.2 of D6.1, a **Conversational Agent** is employed in selected pilot sites to collect information on user preferences and as such, construct a profile of user-centered settings based on aspects, such as the user's role in the manufacturing process, years of experience in that role and articulated preferences. At back-end level, the agent will implement

computation-intensive functions that will support the task of **voice understanding**, i.e., Natural Language Processing (NLP) and more specifically, Automatic Speech Recognition, to process and analyse natural language and translate spoken dialogue to textual data for the user's profile.

4.1.9 Intelligent Marketplace Back-End

The OPTIMAI Intelligent Marketplace has been elaborately presented in D6.3. In that document, the Back-end subsystem is identified as the core of the OPTIMAI Marketplace ecosystem. It provisions all the necessary interfaces to interact with the AI algorithms installed on premises, and supports the automated registration of scrap items. Key functional aspects and features provided by the Intelligent Marketplace Back-end include the following:

- It provides a password-based user authentication and a role-based authorization mechanism.
- It facilitates communication with the Middleware, where the latter will automatically register the defective scrap items to the marketplace through an interface exposed by the Marketplace Back-end Service.
- It receives, or places offerings for sale or purchase the scrap material from different industries.
- It registers AI algorithms in the backend marketplace.
- It manages and reconfigures running AI algorithms in premises.

The high-level functional architecture for the entire OPTIMAI Intelligent Marketplace, as described in D6.3 (including the front-end components) is depicted in Figure 13, below:

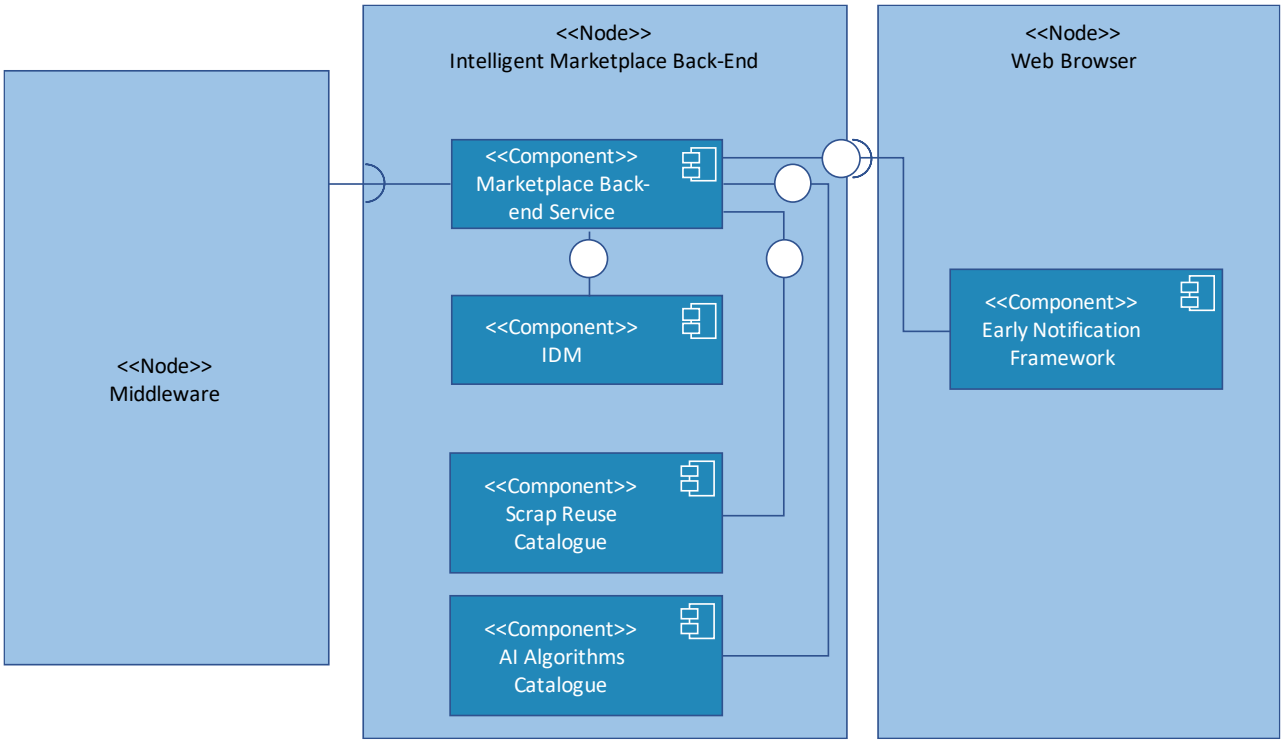


Figure 13. OPTIMAI Intelligent Marketplace component diagram

The components specified in the above diagram are described in the following paragraphs, as well as in Section 4.1.1.33.

4.1.1.24 024 Marketplace Back-end Service

The **Marketplace Back-end Service** component was originally described in Section 2.1 of D6.3. The Back-end service lies at the core of the Back-end subsystem, and is responsible for exposing a REST API (through reverse proxy) that supports the communication of all subsystem components with external entities within the OPTIMAI architectural stack.

4.1.1.25 025 Identity Management System (IDM)

The **IDM** component implements identity and access management, as described in Section 3.2 of D6.3. Its aim will be to control access of users using a single set of login credentials.

> *Additionally, this component will implement Role-Based Access Control (RBAC) in order to restrict access to specific functions provisioned by the Intelligent Marketplace Customer Front-End for authorised users only based on their roles (e.g., seller, buyer). Each user is assigned a role which defines the access level of the user for specific functionality The RBAC shall therefore allow high-level management of access to certain API endpoints (and hence, certain functionalities) only to those users that are meant to be able to use this functionality (e.g., only the authorised seller can have access to a part listing seller's UI).*

Section 4.1.8.2, D2.4, M12.

4.1.1.26 026 AI Algorithm Catalogue

The **AI Algorithm Catalogue** has been defined in Section 3.3 of D6.3. It will be used as a storage repository for AI Algorithms descriptors, which will be published by the Marketplace users identified as AI Algorithm Providers. The descriptor will be used to fill in algorithm details in the Marketplace front-end dashboard, whenever AI algorithms are browsed by Production Line Operator role users in third parties, who aim at purchasing such AI models for increasing their organizations' production quality.

4.1.1.27 027 Scrap Reuse Catalogue

The **Scrap Reuse Catalogue** has been defined in Section 3.4 of D6.3. Its purpose is similar to the AI Algorithm Catalogue, i.e., it is used to store identified defective parts (scrap) into the marketplace, to be purchased by third parties. It will utilize a different descriptor template (see D6.3), so as to list the part in a Scrap Reuse Offering, and make it available through the Marketplace front-end dashboard to the appropriate users.

4.1.10 AI Framework

The AI Framework encapsulates all AI algorithms related to defect detection, prediction, root cause analysis and recommendation of re-configuration settings. It hence combines different algorithmic components that have been described in two of the project's earlier deliverables, namely, D3.10 (**AI-based Production Monitoring Component [AIPMC])** and D6.1 (**Defect Detection and Quality Control Service**). Since in this version of the OPTIMAI architecture, a

choice was made to disentangle AI and DT functionalities, the AI Framework is comprised of the following modules (as described in D2.4):

- *the **Smart Quality Control subsystem** (Section 4.1.11) for driving optimization of the production through data-intensive defect detection and prediction routines.*
- *the **Manufacturing (re-)configuration Service** (Section 4.1.9.1) for the intelligent orchestration of production equipment configuration.*

Section 4.1.9, D2.4, M12.

The high-level functional architecture of the AI Framework is depicted in Figure 14.

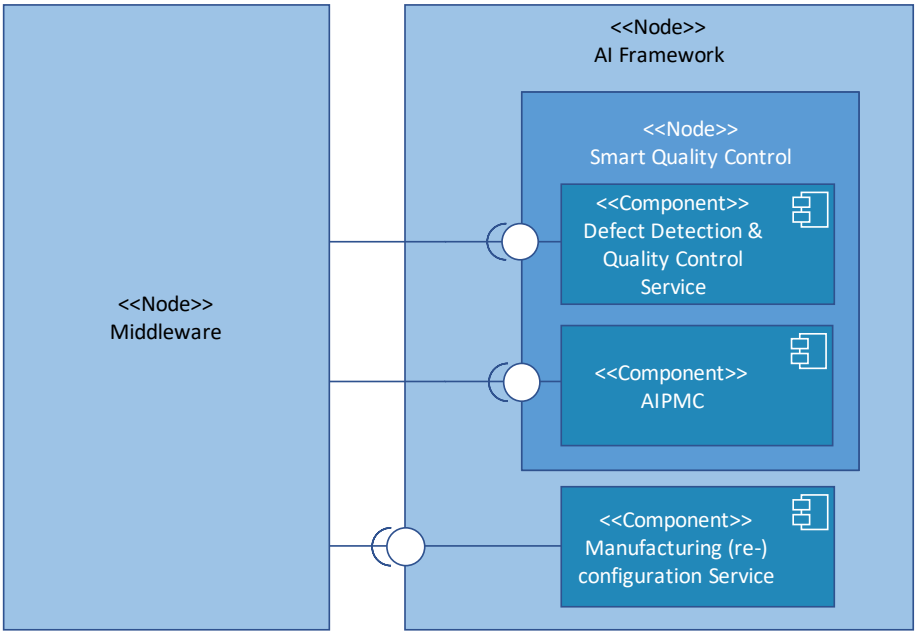


Figure 14. OPTIMAI AI Framework component diagram.

The components comprising the Smart Quality Control subsystem will be elaborated further in Section 4.1.11. The Manufacturing (re-)configuration Service, as not belonging to any subsystem, is described in the next paragraph:

4.1.1.28 028 Manufacturing (re-)configuration Service

The **Manufacturing (re)-configuration Service** has been elaborated in D6.1. Based on the functionality description provided throughout that document, the component embodies a *"reinforcement learning AI agent"*, which is trained on data pertaining to detected anomalies related to suboptimal operations in the production line (AIPMC) and the detection of upcoming and existing defects (Defect Detection and Quality Control Service), and calculates equipment parameters readjustments which realize the automation needed to overcome problematic situations. The calculated parameters can either be applied automatically, or can be forwarded to the OMIDES Back-end, where they are recommended to the appropriate individual, who can then manually apply the proposed reconfiguration.

4.1.11 Smart Quality Control

Void (the contents provided in Section 4.1.11 of D2.4 apply).

4.1.1.29 029 AI-based Production Monitoring Component (AIPMC)

The **AIPMC**, and its functionalities have been extensively described in Section 1.2 of D3.10. Therefore, it will only be briefly re-iterated in this Section. It is related to OPTIMAI's objective of developing AI methodologies for quality control in KLEEMAN, MTCL and TELEVES end-users:

- For the KLEE pilot site, the Hydraulic Power Unit will be monitored to (1) Check parts used in assembly, and (2) Monitor operational data (pressure, speed, noise) to detect suboptimal operation.
- For the MTCL pilot, circuits will be monitored to detect (1) defects during epoxy diffusion, (2) defects from the sawing process.
- For the TVES pilot, the antenna line will be monitored to detect (1) part-defects and (2) assembly errors.

The AIPMC also performs root-cause analysis to suggest cause of defects and suboptimal operation. It therefore executes the following key functionalities:

- It retrieves part and assembly images from the Cloud Data Repository.
- It retrieves timeseries data from the Cloud Data Repository.
- It determines quality status of parts and assemblies.
- It determines operational quality from multisensorial data (timeseries).
- It suggests root-cause of defect/suboptimal operation.
- It submits quality status results to the Middleware.

4.1.1.30 030 Defect Detection and Quality Control Service

As described in brief in D6.1, the **Defect Detection and Quality Control Service** detects upcoming and existing defects. It operates in parallel to the AIPMC, for, where the latter is predisposed with uncovering what causes the production environment to be suboptimal, the Defect Detection and Quality Control Service implements routines that both detect defects in the currently inspected part or manufactured product (e.g., by detecting a mismatch of parts to client order, defects in materials, etc.), as well as predicts upcoming defects (defect analysis) and anomalies (i.e., out-of-the-ordinary measurement values) in the measurements (production monitoring and quality control) with a probability score.

> *Consecutive measurements of previously produced parts are used to extrapolate future measurements, on which the developed defect detection methodologies will be applied for the detection of possible future defects. Several state-of-the-art deep architectures will be examined to support the necessary functionality, such as Deep Residual Networks for defect detection, and Long Short Term Memory (LSTM) and Generative Adversarial Networks (GANs) for defect prediction.*

Section 4.1.11.1, D2.4, M12.

4.1.12 DT Framework

> *The **Digital Twinning Framework** encompasses the different components that comprise simulation routines and functionality of a DT environment. It is based on a modified version of Visual Components **4.4** by VIS, a commercial 3D simulation and visualization solution for*

discrete manufacturing. The solution allows simulating the entire production system at different levels of granularity, from a simple sensor or actuator to a whole manufacturing system with robotics, automation, logistics, and production flows.

Section 4.1.10, D2.4, M12.

Since the simulation is executed on the VC4.4 front-end solution, the components described in the following sub-Section constitute the base for the simulation (the digital model), which is the virtual representation of the physical assets in the virtual simulation space.

4.1.1.31 031 DT Process Models

Void (the contents provided in Section 4.1.10.1 of D2.4 apply).

4.1.1.32 032 Virtualised Sensor Network

Void (the contents provided in Section 4.1.10.2 of D2.4 apply).

4.1.13 End-users' Applications

4.1.1.33 033 OPTIMAI Intelligent Marketplace Dashboard

The **OPTIMAI Intelligent Marketplace Dashboard** has been described in Section 3.1 of D6.3. It constitutes a unified Single Page Application (SPA), which delivers a UI experience for users to browse through Scrap Reuse Offerings and AI Algorithms. It supports login and registration functionality to identify users as belonging to one of four basic stakeholders that will access the OPTIMAI Marketplace, i.e., the Administrator, the Product Line Operator, the AI Algorithm Provider and the Scrap Trader. Four dashboard views will expose different functionalities for each user type.

*As has previously been mentioned in Section **4.1.9**, this component will communicate with the Intelligent Marketplace Back-End for provisioning of the different services needed to satisfy user requests.*

Section 4.1.12.1, D2.4, M12.

4.1.1.34 034 Visual Components Software

***The Visual Components Software will be a custom offering based on the VC4.4** commercial solution by VIS, and is aimed at realistically representing the DT using a rich 3D graphics engine, while also exposing graphical UIs with bindings to the Digital Twinning subsystem components for e.g., creating a virtual factory layout, or defining processes based on tasks. In addition, a statistics UI available in VC **4.4** will be provisioned, enabling a user to monitor the performance of the virtual factory according to the layout configuration created.*

Section 4.1.12.2, D2.4, M12.

4.1.14 OMIDES Front-End application

The **OMIDES Front-End** is thoroughly explored in D6.1. It comprises a cross-platform (i.e., PC or tablet) solution for supporting operators on the shop floor during the execution of their daily tasks, with the intention being to optimize production and increase their efficiency. Facilitating

communication with the OMIDES Back-end (Section 4.1.8), it presents a visual environment for both:

- Informs the user on the results of defect detection and the status of the production environment based on the outputs of the AI Framework,
- Facilitates alerts to users to perform re-configurations manually on the appropriate device, thus allowing for fast response depending on the current context (Early notification framework).

The application comprises three subsystems, described in the following paragraphs.

4.1.1.35 035 Early Notification Framework

The **Early Notification Framework** is responsible for notifying the appropriate person in the appropriate device for abnormal operations, predicted and detected defects so as to facilitate an immediate (manual) response. Such notifications will be triggered by defect detection and measurements anomalies detection events occurring in the Smart Quality Control (see Section 4.1.11). The design of the notifications content is addressed in Section 2.3 of D6.1, while its visual output is described in Section 4.4 of that same report.

4.1.1.36 036 Visual Analytics

The **Visual Analytics** components of the OMIDES Front-end application encompass GUI components presenting the analysed data visualizations to support operators and production managers to make more time-consuming decisions to optimize the production in the long run. Following best practices for responsive web applications' design, and integrating well established user experience principles (e.g., usability), it comprises a set of graph widgets and input items, allowing users to tailor the visualization environments to their needs. These components are described, and high-fidelity mock-ups are depicted in Section 4 of D6.1.

4.1.1.37 037 Conversational Agent Interface (CAI)

The **Conversational Agent Interface (CAI)** embodies an AI-based front-end tool for enabling interaction of the user with the OMIDES Front-end, which will determine user preferences by engaging in vocal dialogue with the user. Since the computation-intensive functions will occur at the OMIDES Back-end (see Section 4.1.1.23), the CAI will incorporate functions for capturing the users' voice input, manage the dialogue flow (i.e., selecting which questions to ask), and synthesize speech to respond to users' remarks in an audible format. The agent front-end has been described in more detail in Section 5.2 of D6.1.

4.1.15 OPTIMAI Augmented Reality solution

Since its inception, OPTIMAI has envisioned Augmented Reality (AR) as a key enabler to achieve the project's ambitious goals. Throughout the first 18 months of the project, as the designs for the different system components matured, it became evident that the AR ecosystem contemplated (based on a custom smart-glasses hardware solution provisioned by YBQ) constitutes a standalone component comprising a multimodal pipeline for alerting the operators on the shop floor about detected or predicted defects of the manufacturing pipeline providing them intuitive ways for rapid response towards resolving the problem. This component will

operate in tandem with the OMIDES Front-end providing alternative workflows based on the OPTIMAI's smart glasses ecosystem. The aforementioned endeavour is materialized by the MAIAR (optiMAI AR) Software, which provides pervasive visualization, recommendation and interpretation directly on top of the smart glasses hardware.

4.1.1.38 038 MAIAR Software

As a standalone module, the MAIAR software encapsulates many of the functionalities supported in the OMIDES Front-end, albeit contextualised to support operation in an AR environment. This means that the MAIAR software integrates its own separate design for the visualization of defect and quality results algorithmic outputs, the adaptation of the visualisation environment triggered by an integrated ontological decision-making component, and support for hand gesture-based interaction to receive input from the smart glasses wearer.

The high-level functional architecture of the MAIAR Software is depicted in Figure 15. An elaborate description of the various elements comprising the application will be delivered in D5.3.

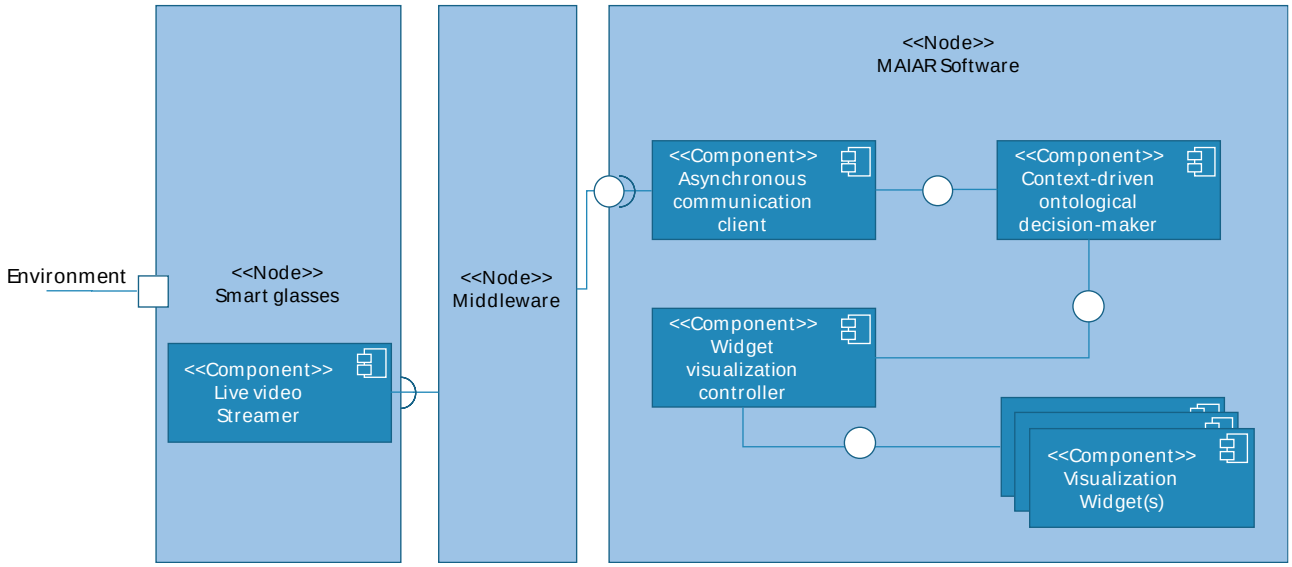


Figure 15. MAIAR Software component diagram.

4.2 OPTIMAI alignment to standards-led reference architectures

Void (the contents provided in Section 4.2 of D2.4 apply).

4.2.1 Alignment to RAMI 4.0

RAMI 4.0, as specified in Section 2.2.1, is a cubic map intended to encapsulate the entirety of concepts and elements that make up the modern I4.0 smart factory environment. It delivers no concrete architecture or implementation guidelines, but rather, presents a structured approach to address the particulars of a given smart factory use case.

Interpreting OPTIMAI in the context of RAMI4.0, the foreseen UCs to be evaluated in the three pilot sites (and thus, the defined architecture for the proposed solution to those UCs' requirements) are:

- *Reducing the number of quality defects in the production line ("Zero defect quality inspection" – UC1).*
- *Improving the efficiency of the production line by optimally calibrating machines/robotic cells in a way that decreases stoppages ("Production line setup-calibration" – UC2).*
- *Optimising the production of the manufacturing line by means of a digital twin where optimal product manufacturing sequence can be calculated for future planning purposes ("Production planning" – UC3).*

*It becomes apparent that these processes are created inside the 'Instance' phase of the Life Cycle & Value Stream axis defined in RAMI 4.0. Particularly, the processes described in the 24 sub-UCs defined in D2.6 fall within the 'Production' state, as the actions undertaken to deal with quality issues during manufacturing execution, and formulating knowledge for ideally setting up the manufacturing environment. Hence, with the placement in one of the three axes specified, OPTIMAI can be layered on top of a 2D slice extracted from the cubic map (**Figure 16**) presenting a layer-and-hierarchy mapping in terms of "Production".*

Section 4.2.1, D2.4, M12.

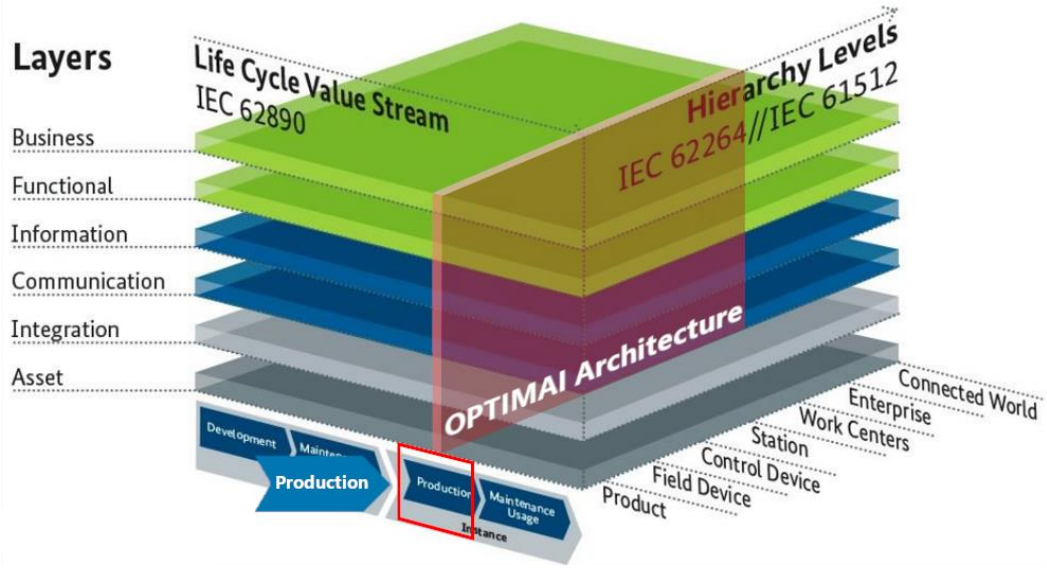


Figure 16: OPTIMAI placement within the RAMI 4.0 cubic model. Adapted from the original Graphic © Plattform Industrie 4.0 and ZVEI, retrieved from [4] (reproduced here from D2.4, M12).

*Based on the described operations of the functional components presented in the previous Section, the OPTIMAI architecture is mapped onto the 2D layer-and-hierarchy slice as shown in **Figure 17**, thus enabling the compatibility with the RAMI 4.0 guiding principles to be better illustrated.*

As can be seen, OPTIMAI defines components across all Layers and Hierarchy Levels for the "Production" state of the 'Instance' Life Cycle & Value Stream Phase:

Section 4.2.1, D2.4, M12.

- The QCS Network is comprised of sensory apparatus and modules installed on the shop floor for collection of data and actuation. All QCS Network blocks therefore can be mapped to the 'Asset' RAMI 4.0 Layer. Based on the extent of intelligence that can be

applied to the devices (i.e., simply generating and propagating values, or actually executing intelligent functions to produce higher-order data and actuation commands), defined modules are mapped to both the 'Field Device' (001) and 'Control Device' (002, 003) Hierarchy Levels.

- *The Middleware subsystem (components **004-007**) similarly maps to the 'Integration' Layer as it underpins the communication of the entities in the real world with the higher-level software components of the architecture. Dealing heavily with the exchange of data among functional units, sensors and subsystems, the Middleware components are aligned to the functional areas defined within the responsibility of a MES, e.g., the 'Work Centers' Hierarchy Level, with the added benefit of implementing security mechanisms as early as the data acquisition process (i.e., as soon as data enters the system).*

Section 4.2.1, D2.4, M12.

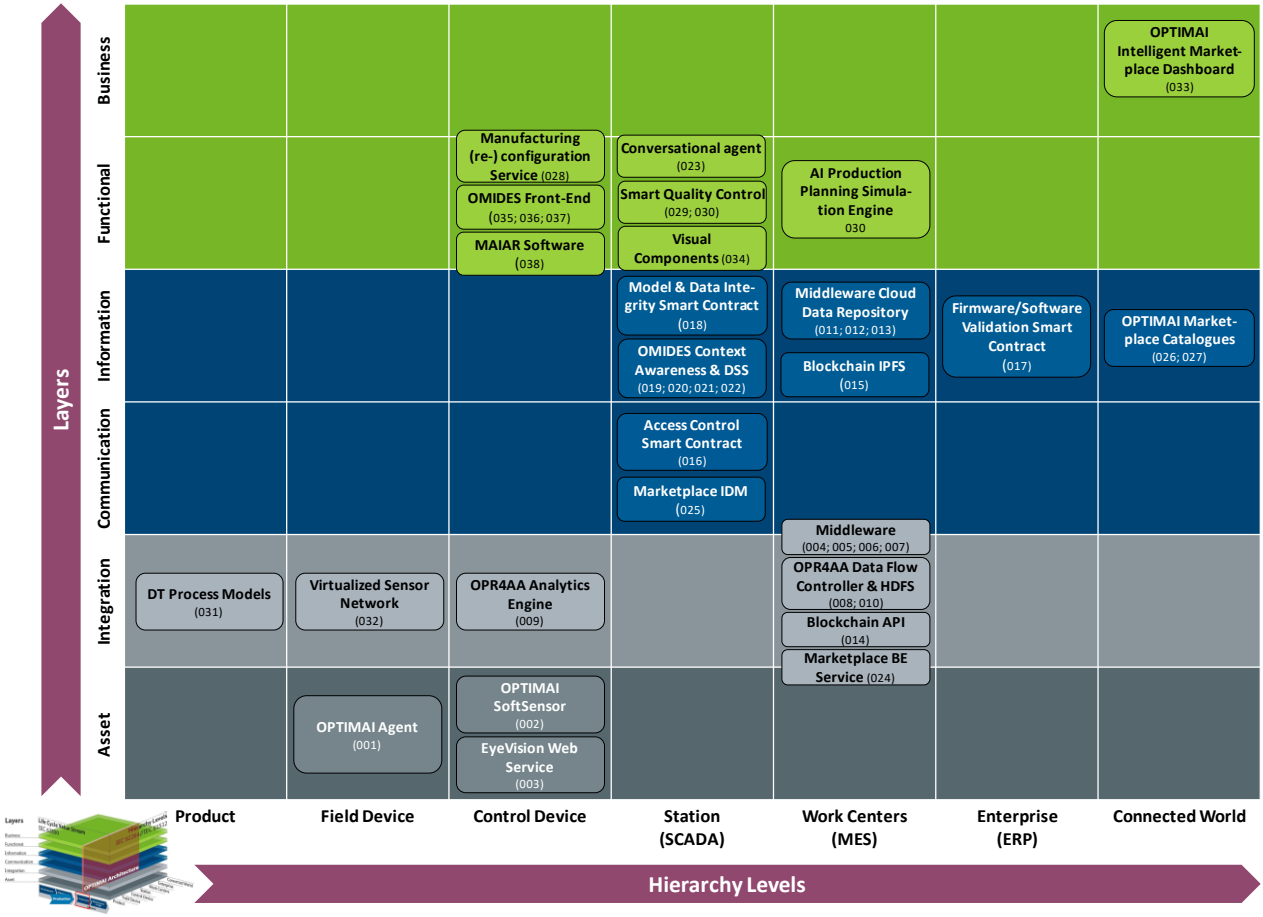


Figure 17: OPTIMAI Layer-and-Hierarchy mapping to RAMI 4.0.

- Components of the OPR4AA Platform facilitate the edge-based calculation and delivery of control parameters in an attempt to optimize acquisition and actuation. As specified in D2.4, edge-based computations align to the 'Integration' RAMI 4.0 Layer, facilitating a direct interaction of readings from the QCS Network with edge-based AI models through the Middleware. On the Hierarchy Levels axis, the Data flow controller (008) maps to the 'Work Centers' Hierarchy Level, since its core function is to manage data flow among the

other components of the subsystem. The Analytics engine (009) functions in the same capacity as the QCS Network intelligent components at the 'Control Device' Hierarchy Level. Finally, the HDFS (010) encompasses collection and storage of data, which occurs also in the 'Work Centers' Hierarchy Level.

- *The Middleware Cloud Data Repository (blocks **011-013**) complements the aforementioned functional area of MES regarding collection and storage of process and production data occurring within the 'Work Centers' Hierarchy Level, while mapping to the 'Information' Layer where RAMI 4.0 considers structured data storage.*

Section 4.2.1, D2.4, M12.

- With respect to the Blockchain Framework, the Blockchain API Service (014) functions in a similar capacity to the OPR4AA Data flow controller at the intersection of the 'Integration' Layer and 'Work Centers' Hierarchy Level. the IPFS (015) is defined within the 'Information' Layer, for *"dealing with the distributed ledger architecture (e.g., a database spread across locations) for storing records"* (D2.4). The smart contract components (016-018) are mapped according to the D2.4 established guidelines:

*[…] validation of firmware and software installed on sensors and OPTIMAI middleware (017) relates to functional capacities of ERP systems to provide unified and centralised device management (aligning 017 to the 'Enterprise' Hierarchy Level), while checks on data integrity are a crucial component in the data acquisition processes occurring at SCADA systems [18] (placing 018 […] within the 'Station' Hierarchy Level). Finally, Access Control functionality refers to a cybersecurity perspective of dealing with security challenges related to attack vectors to communication methods over the network [17], particularly relevant to the data acquisition part of modern SCADA systems. Hence, functional block **016** is mapped at the intersection of the 'Communication' Layer with the 'Station' Hierarchy Level.*

Section 4.2.1, D2.4, M12.

- The OMIDES Back-End components support SCADA workflows occurring at a UI or HMI terminal, therefore mapping components 019-023 to the 'Station' RAMI 4.0 Hierarchy Level. On the Layers axis, the DSS Engine (022) implements storage and data flow functions (e.g., forwarding of alerts and notification to the proper user accounts), which map the component to the 'Information' Layer. The Conversational Agent Back-end is used to determine adaptive properties of UIs and HMIs, thus mapping 023 to the 'Functional' layer. Components 019-021 relate to processing of information for determining higher-order information (i.e., context) on raw data, mapping them to the 'Information' Layer.
- The Intelligent Marketplace Back-End provides functionalities at the intersection of the 'Integration' Layer and 'Work Centers' Hierarchy Level (024) for regulating flow of data inside the Marketplace ecosystem, as well as access control functionality (025), which, as previously mentioned, maps at the intersection of the 'Communication' Layer with the 'Station' Hierarchy Level. The two catalogues (026-027) deal with storage (hence, mapping

to the 'Information' Layer), albeit intended to *"provide a connection between the factory and the outside world (third parties) thus mapping clearly to the 'Connected World' RAMI 4.0 Hierarchy Level extension"* (D2.4).

- Within the AI Framework, components maintain the mappings established in D2.4:

  - *[…] the Manufacturing (re-)configuration Service component (**028**) […] deliver**s** decision-making on the information produced by **the Smart Quality Control** toward applying automated and manual re configurations of individual machine parameters. **The module is therefore found at the 'Functional' Level, and relates** to processes described in the 'Control Device' Hierarchy Level.*
  - *[…] Smart Quality Control components (**029** & **030**) perform the necessary processing checks and deliver decision support toward defect identification and detection events notification based on real-time data generated by other modules in the OPTIMAI architecture. They are hence a crucial component to the 'brain' of the OPTIMAI solution, and are defined at the intersection of 'Functional' Layer and 'Station' Hierarchy Level.*

  Section 4.2.1, D2.4, M12.

- Similarly, DT Framework components maintain their mappings in the 'Integration' Layer, as established in D2.4, identified as either "Engineering" or "Runtime" data respectively within the GDTA:

  - ***DT Process Models (031)** refers to the concept of "Engineering" Data, i.e., topological information about the production plant that should be taken into account as they can affect production parameters (such as the distance a worker has to travel from one area to the next in a manual workstation). Hence, we can define those data as originating in the 'Product' Level, referring to the production facilities and the interdependencies they impose [5].*
  - *The Virtualised Sensor Network (**032**) supplements the functionality provisioned by the QCS Network, similarly placing it within the 'Field Device' Level.*

  Section 4.2.1, D2.4, M12.

- With respect to the End-users' Applications, these also maintain their established mappings from D2.4:

  - *The **OPTIMAI Intelligent Marketplace Dashboard** (033), provides a connection between the factory and the outside world (third parties) thus mapping clearly to the 'Connected World' RAMI 4.0 Hierarchy Level extension. It becomes further apparent that block 033 refers to an application designed to influence decisions at the strategic level (such as the acquisition/reuse of scrap produced by another organisation), while the Back-End subsystem deals with recording of transactional information and feedback. They are thus found within the 'Business' and 'Information' Layers respectively.*
  - *The **Visual Components Software** (034) represents a rich UI provisioned for the supervisory control part of SCADA systems, which is aimed at enabling monitoring of real-time data generated by the simulation and the direct interaction with the DT devices and sensors through emulated HMI software. Hence, for OPTIMAI purposes, this component is found at the 'Station' Level.*

- *The OMIDES Front-End on the other hand (functional blocks **035-037**), combines with the Manufacturing (re-)configuration Service component (**028**) to deliver decision-making on the information produced by **the Smart Quality Control**, toward applying automated and manual re configurations of individual machine parameters. These modules are therefore found at the 'Functional' Level, and relate to processes described in the 'Control Device' Hierarchy Level.*

Section 4.2.1, D2.4, M12.

- Finally, the MAIAR Software encapsulates functionalities that closely align to those of the OMIDES Front-end and Back-end modules. At a high-level, we will map the MAIAR Software at the intersection of the 'Functional' Level, and the processes described in the 'Control Device' Hierarchy Level.

4.2.2 Alignment to IIRA

As has been mentioned in Section 2, IIRA and RAMI 4.0 are significantly similar in their support of SOAs, i.e., decomposition of system functionality into an array of interconnected services. Because of the service-oriented approach followed in the OPTIMAI architecture, as well as its overt alignment to RAMI 4.0 described in the previous Section, by extension OPTIMAI can significantly be parallelised to the IIRA as well. Regarding the IIRA Viewpoints, with respect to the OPTIMAI project, the following are thoroughly defined in the context of OPTIMAI:

- *The Usage Viewpoint of the OPTIMAI architecture is described in detail in deliverable D2.6, which precedes the present document in specifying the expected usage of the OPTIMAI system. In specifying the use cases in detail prior to the architecture (i.e., taking use case description into account when designing the ICT systems to support each case), OPTIMAI aligns to the principle of IIRA for top Viewpoints to guide the design of the viewpoint directly below.*
- *The Functional Viewpoint, which is reflected in the specification of the OPTIMAI system functional blocks and their correspondences to roles and responsibilities (Section 4.1), related to the functions they are expected to perform to support the use cases. As IIRA and RAMI 4.0 intersect at the IIoT systems for smart manufacturing domain, RA alignment principles are established for the functional mapping of the IIRA and RAMI 4.0 [8], which are iterated in **Figure 18**. Based on these mappings, correspondences between the OPTIMAI architecture (from an IIoT solution perspective) and the IIRA are shown in **Figure 19**.*

Section 4.2.2, D2.4, M12.

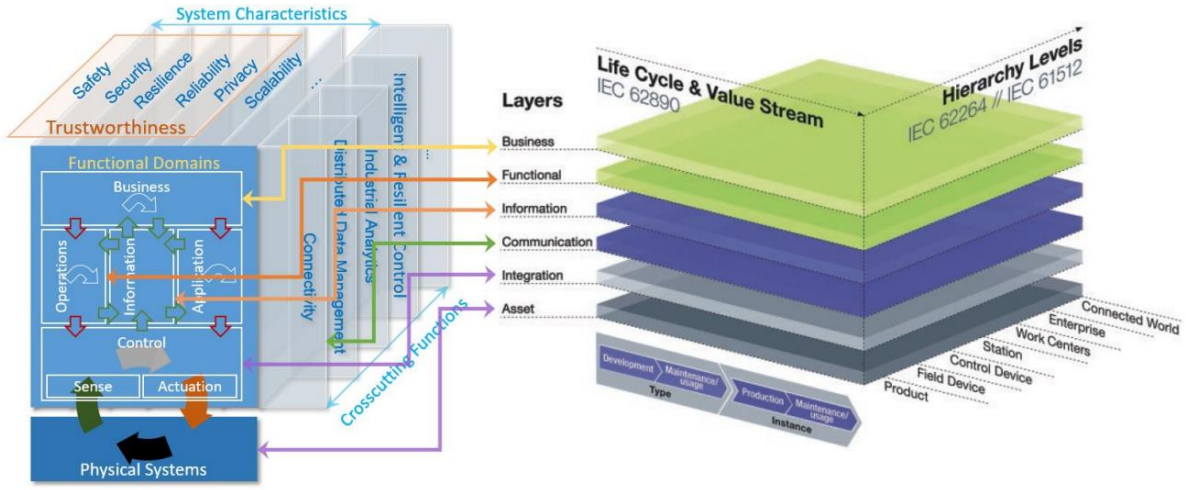


Figure 18: mapping between the IIRA Functional Viewpoint and IT layers established in RAMI 4.0. Source: [8] (retrieved from D2.4, M12).

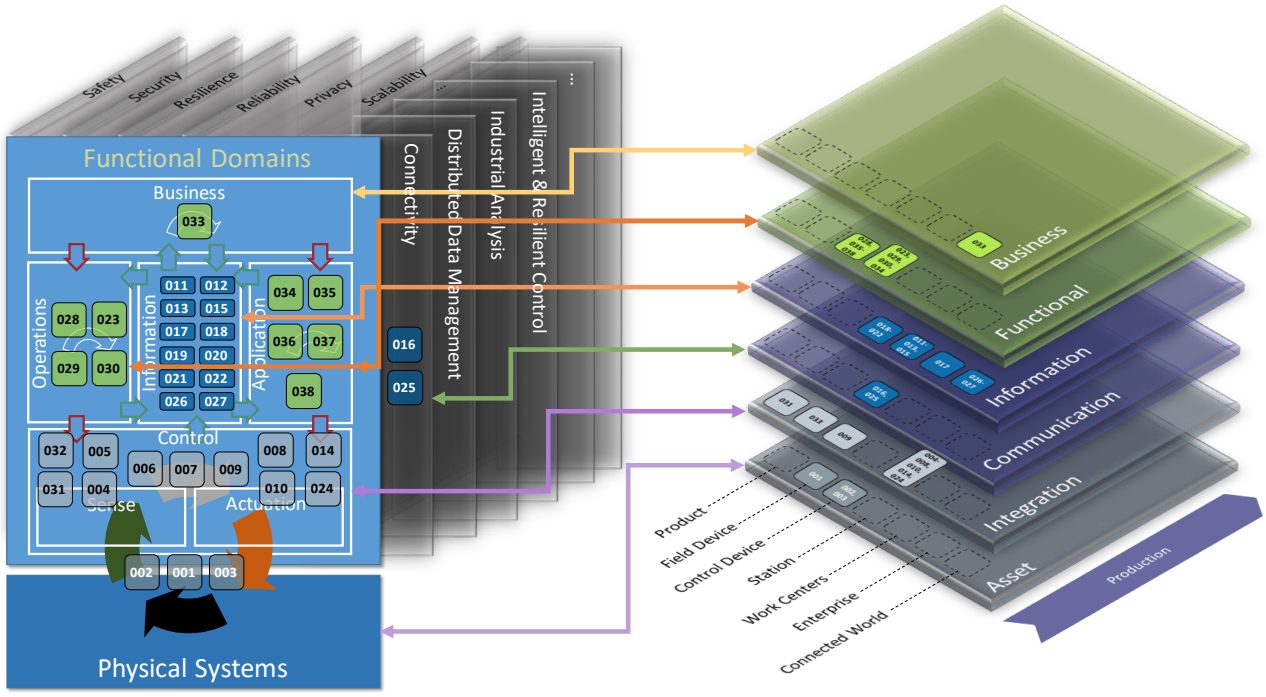


Figure 19: Functional mapping of the OPTIMAI Architecture to IIRA based on the alignment to RAMI 4.0.

Following the guidelines presented in Figure 19, the OPTIMAI architectural components can be mapped to the following IIRA *Functional Domains*, as indicated in the following Table (Table 4):

Table 4: Mapping between IIRA Functional Domains and OPTIMAI functional components.

IIRA Functional Domain	Description	OPTIMAI Functional Blocks
Physical Systems	Despite not being formally established as a Functional Domain in IIRA, Physical systems are directly mapped onto the RAMI 4.0 Assets Layer, and are understood as the physical resources on the	**001, 002, 003**

	factory shop floor. Because of dealing with sensing and actuation, these assets can also be mapped to the Control domain, as elaborated below.	
Control domain	Includes components, whose functions deal mainly with the control, sensing and actuation on the physical systems. Such functions involve the collection of data from sensors, potential application of logic and eventual execution of actuation commands. Because of this property, within the IIRA, these assets are considered to be in close "proximity to the physical systems they control" [7] (i.e., the Network Edge). Identifies the following functions: • **Sensing:** Read data from sensors. • **Actuation:** Apply configuration to a hardware component. • **Communication:** Connect to external entities. • **Entity abstraction:** Sensor digital representation (digital twin). • **Modelling:** Monitoring of operation. • **Asset management:** Exercises management over the hardware. • **Executor:** Exercises logic to facilitate control (at the edge-level).	Sensing **001; 002; 003; 005; 008** Actuation **001; 003** Communication **004; 007; 008; 014; 024** Entity Abstraction **031, 032** Modelling **006; 009; 010** Asset Management **003; 009; 010** Executor **003; 009; 010**
Operations domain	Exercises monitoring, management and control over the assets in the Control domain. Essentially, this domain deals with operations regarding decision-making based on data capturing, processing and validation. Hence, it is directly mapped to the functions defined within the RAMI 4.0 Functional Layer. Identifies the following functions: • **Provisioning and Deployment:** Configure and deploy/retire assets from the network • **Asset Management:** Intelligently manage assets' processes. • **Monitoring & Diagnostics:** Detection and prediction of issues. • **Prognostics:** Predictive analytics. • **Optimisation:** Improve asset performance.	Provisioning & Deployment **028** Asset Management **023; 028** Monitoring & Diagnostics **029; 030** Prognostics **030** Optimisation **028**

Information domain	Deals with the management, processing, transformation and storage of data. Maps directly to the RAMI 4.0 Information Layer. Identifies the following functions: • **Analytics:** Data modelling and rules' application. • **Data:** Pre-processing and persistence of data.	Analytics **017; 018; 019; 020; 021; 022** Data **011; 012; 013; 015; 026; 027**
Application domain	Deals with functions that support application-specific logic. Within the IIRA it differentiates from the Operations domain by being use-case-specific, yet, its functions (listed below) maintain mapping to the RAMI 4.0 Functional Layer. • **Logic and Rules:** implements specific rules and models tied to a specific use case. • **APIs and UI:** Refers to the interfaces exposed towards other functional entities, including human operators.	Logic & Rules **034; 035; 038** APIs & UI **036, 037, 038**
Business domain	Deals with functions that implement business processes. Maps directly onto the RAMI 4.0 Business Layer.	**033**

*Not depicted in the above Table, the RAMI 4.0 Communication Layer (in which OPTIMAI component**s 016 and 025 are** defined) is mapped onto the Connectivity Crosscutting Function defined in IIRA, reflecting the need for a particular security function to be implemented across communications of the functional components [7] (in OPTIMAI's case, **Role-Based** Access Control).*

Finally, some elements related to the Implementation viewpoint can be seen in the descriptions of both the technologies' selected technical components (e.g., sensor hardware) and communication schemes (e.g., OPC-UA), while the strategic goals and benefits driving the system implementation and deployment across the three industrial pilot sites (i.e., the IIRA Business Viewpoint) are reflected in the Key Performance Indicators (KPIs) elaborated for each case in D2.6.

Section 4.2.2, D2.4, M12.

4.2.3 Key takeaways

Void (the contents provided in Section 4.2.3 of D2.4 apply).

5 OPTIMAI Architecture - Information view

Void (the contents provided in Section 5 of D2.4 apply).

5.1 Overview information flow

In particular, the purpose of the OPTIMAI information view is to specify details with regard to the communication channels and data flows between different components, taking into account the input, output, and dependencies documented for each one.

> ***Figure 20*** *presents an overview of the OPTIMAI Information Flow Diagram (IFD) layered on top of the functional architectural viewpoint, hence depicting the manner in which information flows through the system between locations, ether within the same, or between federated computer systems. The goal is to define the manner in which information is exchanged between OPTIMAI functional blocks, as well as with potential external entities (e.g., user equipment). All foreseen data exchanges are represented following the producer-consumer paradigm [20], in which any one of the interdependent functional blocks can assume the role of a producer (of data), a consumer, or both.*

Section 5.1, D2.4, M12.

Following this paradigm, concrete interfaces between the functional components have been identified, and they reveal the data that a "producer" block generates and sends to its interdependent "consumer". The following paragraphs provide a thorough description of the interfaces (presenting the message exchange formats, request and response schemas, etc.). Therefore, the content of the current section offers a comprehensive reference frame for the final system implementation.

A template created for the purpose of interface definition was completed by all developers/component owners to capture message exchange information for each interface identified among theirs and other components in the architecture (Figure 20). This template is presented in pages 7-15 of the Component refinement template included in Appendix A. The input provided by partners contributed to the identification of the specific inter-exchanges listed in Figure 20. For the various parameters defined in these exchanges, the information mentioned in Table 5 will be specified.

Communication of the various functional modules and components was hence defined in accordance to the information collected through the component refinement template (Appendix A). These interfaces allow both synchronous and asynchronous communications between the architectural components.

Synchronous communication in the framework of the defined interfaces refers to a mode of communication that is based on a request-response API mechanism. For example, a module constructs a request (HTTP) structure, sends the request and the interdependent entity

processes the request and sends back a response. All the synchronous interfaces in the OPTIMAI Information view are implemented using REST clients.

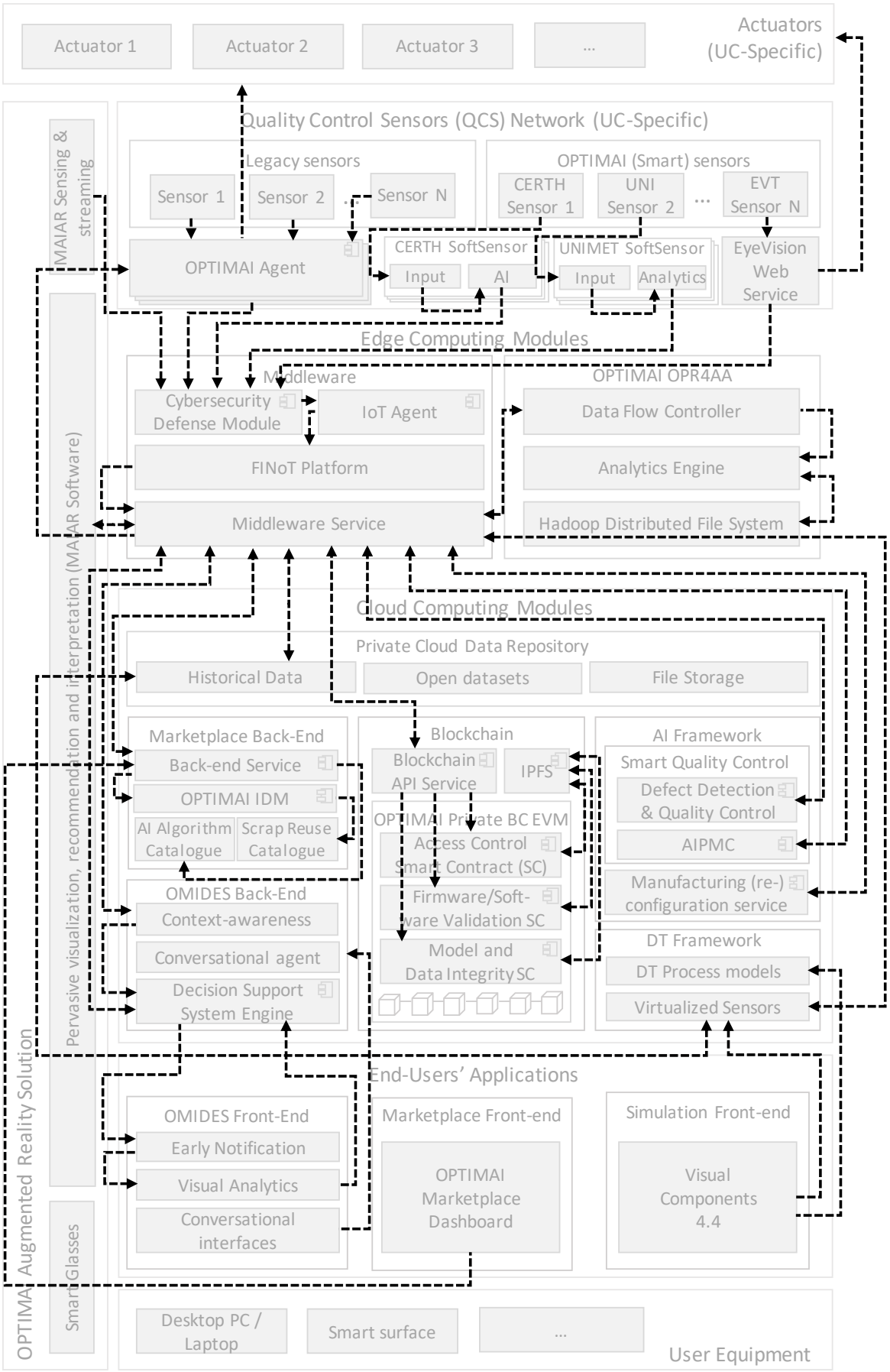


Figure 20: OPTIMAI high-level Information Flow Diagram.

Table 5: Interface information elements description and conventions used.

Information element	Description and notations
Parameter	Parameter name.
Requisitness	Indicates requisiteness of the parameter, *e.g.*, whether the parameter is: (M) mandatory; (O) optional; (CM) conditional mandatory; (CO) conditional optional.
Cardinality	Indicates cardinality, *e.g.*, number of allowed appearances of this parameter in the message payload. It may indicate a range of cardinality, as well as imply requisitness (*e.g.*, optional parameters may have a cardinality of 0).
Type	The type of data communicated by this parameter (*e.g.*, string, integer, *etc.*)
Description	A short description of the parameter.

In the **asynchronous** communication mode, components communicate through a publish/subscribe paradigm relying on a message broker component. In the asynchronous case, a component constructs a request (HTTP) structure, sends the request, but doesn't wait for a response, as opposed to synchronous communication. Within OPTIMAI, this mode of communication is realised by utilising the OASIS MQTT protocol, which is implemented using the project's Middleware solution[3]. Basically, a client sends messages to a server (broker) over the middleware. These messages are published to a specific address (an exchange or queue) where they can be consumed by clients. Queues retain a copy of each message internally until the client receives and processes this message. A unique queue can be assigned to each client, allowing clients to consume individual copies of messages at their own pace, without affecting the functionality of other clients. The Middleware is therefore be used to manage distribution to the queues so that publishers can add messages to the queues and subscribers can consume them.

The communication interfaces for each component are, or will be described in detail in the component's respective deliverable.

5.2 Use Case-specific information flow

*This section presents **selected** indicative **application scenarios of the OPTIMAI system in the form of sequence diagrams**, aimed at clarifying how information flows through the system so as to enable specific functionality pertinent to the three use case types defined for each industrial pilot site in the project. These **sequence diagrams** are meant to elaborate on the high level overview of the flow of information presented in the previous paragraph, and provide a simple, yet detailed overview of the overall synergies and interdependent component*

[3] Alternatively, RabbitMQ can be used, because it is a mature message broker providing support for a straightforward client and server implementation of the MQTT protocol, enabling the distribution of messages through a publish/subscribe messaging pattern and the creation of work queues.

flows created for individual process delivering on specific use-case functionalities. Each diagram is carefully designed to demonstrate the roles and responsibilities of the different components in the OPTIMAI architecture with respect to the use case in question. We omit to include a description of the UCs themselves, as this information is available in deliverable D2.6.

Section 5.2, D2.4, M12.

In specific, 8 application scenarios of the system have been elected, since they represent key functionality offered by the system in the different pilot application UCs. The choice of scenarios to be elaborated as sequence diagrams has been made so as to incorporate, and hence, demonstrate indicative functionalities and information flow involving all of the main components and modules of the OPTIMAI architectural stack.

The sequence diagrams have been designed taking into account the industrial partners' requirements, information contributed by the technical partners using the component refinement template (Appendix A), information retrieved from the technical deliverables in WPs 3-6, and finally, discussion with all partners during the final architecture workshop, where the information was presented as use case-specific Information Flow Diagrams (IFDs, see Figure 20 for the overall IFD) and was agreed upon by all Consortium partners.

5.2.1 Multi-sensory data acquisition

The diagram presented in Figure 21 showcases the different ways in which measurements are entered into the system and are forwarded to the Middleware for externalization to the other subsystems and components of the OPTIMAI platform. Three distinct cases of data acquisition are represented, i.e., data coming from existing (legacy) sensors via the OPTIMAI Agent; higher-order measurement generation from OPTIMAI-developed SoftSensors; and direct propagation of measurements from "smart" sensors with a direct connection to the Middleware (i.e., EVT sensors). The optional scenario of the EyeVision Web Service being used in the capacity of a SoftSensor is also displayed. The Middleware applies cyber security at the data acquisition level, executes CEP and storage and eventually distributes the data over the appropriate interface.

The application scenario is relevant in both UC1 and UC2, as defined in D2.6.

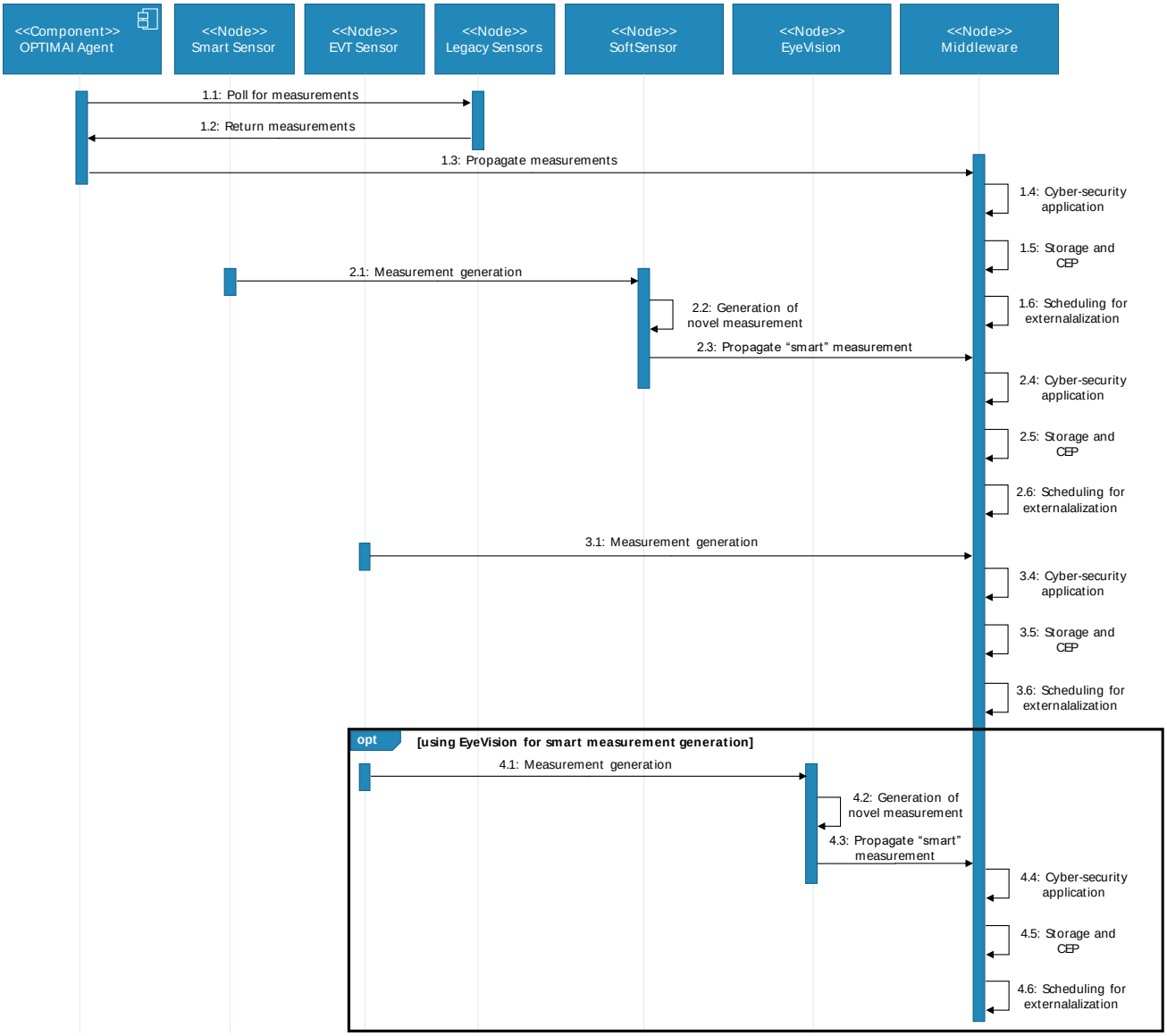


Figure 21: OPTIMAI multi-sensory data acquisition sequence diagram.

5.2.2 Time-critical configurations: bypassing the Middleware

The diagram presented in Figure 22 demonstrates a particular scenario presented in Section 5.4 of D3.1. In this case, granted that for time critical, minor adjustments that can be handled by extremely lightweight algorithms, passing the data through the Middleware might be an overkill solution, the architecture can support execution directly on the shop floor. This is enabled through the EyeVision Web Service exposing a C++ plugin interface, enabling execution of algorithms on the shop floor device on which it is running. Depending on the actuator hardware and software configuration, actuation commands can be applied directly through the EyeVision software, or by means of the OPTIMAI Agent.

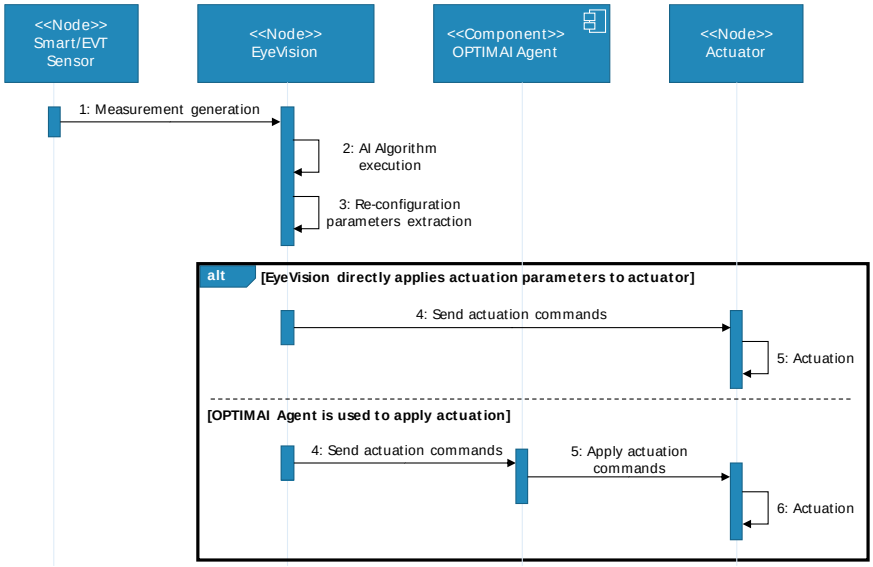


Figure 22: OPTIMAI time-critical configuration and bypassing the Middleware sequence diagram.

This application scenario is relevant to UC2.

5.2.3 Defect detection and production line monitoring

The diagram in Figure 23 presents the use of the system in the capacity described in Section 4 of D6.1, demonstrating the use of the AI Framework to detect events through the Defect Detection and Quality Control Service and the AIPMC (executed in parallel). The results are then visualized to a production operator using the OMIDES on a desktop or tablet terminal device. The case in which data needs to be analysed for context (e.g., whenever data from the wearable camera sensor on the OPTIMAI Smart Glasses is processed), the triggering of the OMIDES Back end modules is laid out. The user can interact with the OMIDES Front-end components and receive information in a comprehensive manner through the various visualization widgets. The application scenario is related to all UCs.

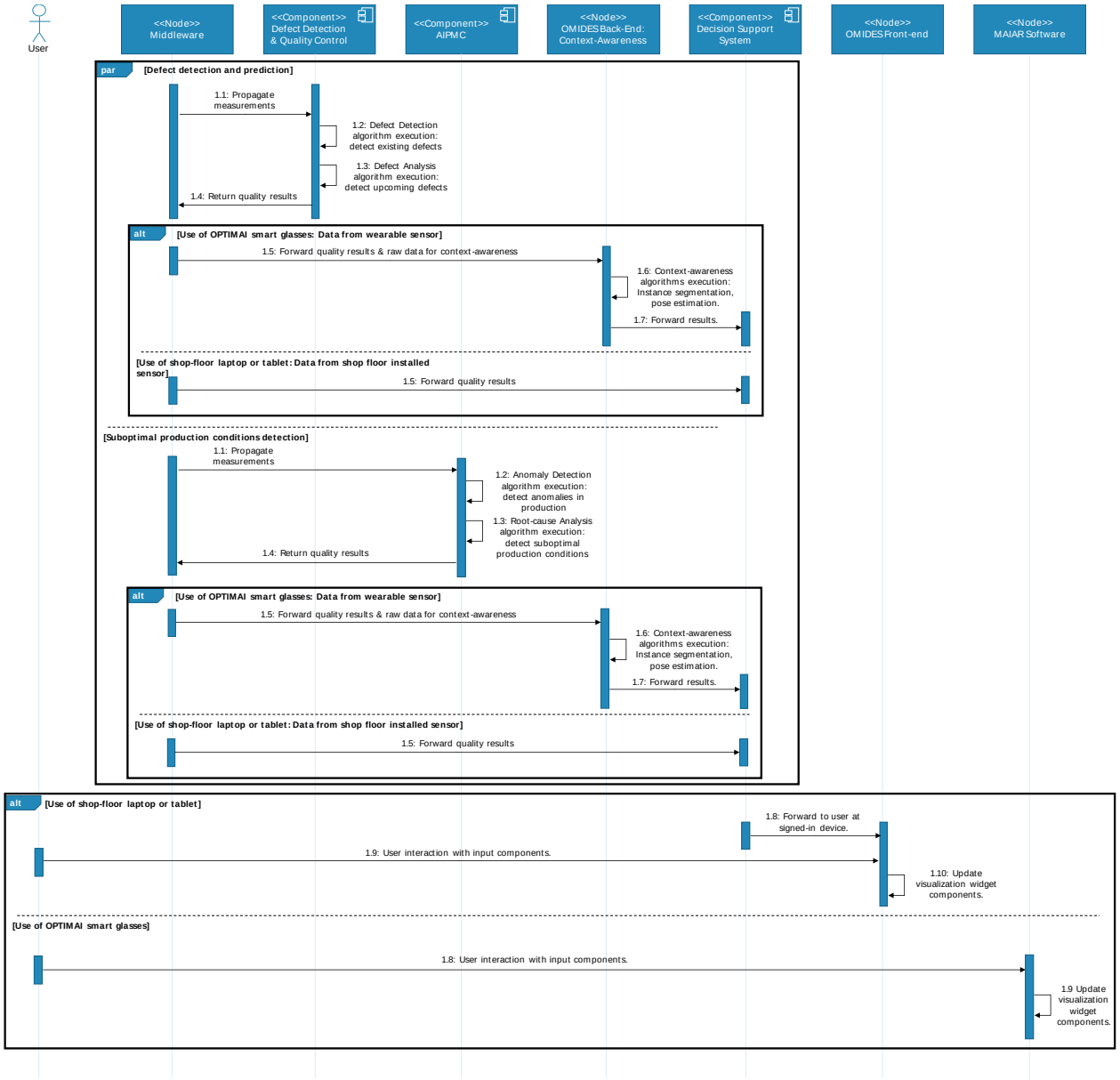


Figure 23: Defect detection and production line monitoring sequence diagram.

5.2.4 On-the-edge processing

The diagram in Figure 24 pertains to the capability of the system to utilise on-the-edge algorithm execution through the OPR4AA Platform. The diagram demonstrates how data is entered to the OPR4AA Platform by means of the Data Flow Controller, along with the interworking of the Analytics Engine (Apache Spark) and HDFS (Hadoop) pipelines. In this way, measurements data can be entered into the OPR4AA Platform and quality results can be obtained bypassing the AI Framework (in cases where the algorithmic execution is lightweight enough to allow task execution on the edge). Optionally, algorithms for the re-configuration of software on the actuators can run in the same manner, with automatic deployment of actuation commands being possible by means of the OPTIMAI Agent.

The application scenario is relevant in both UC1 and UC2.

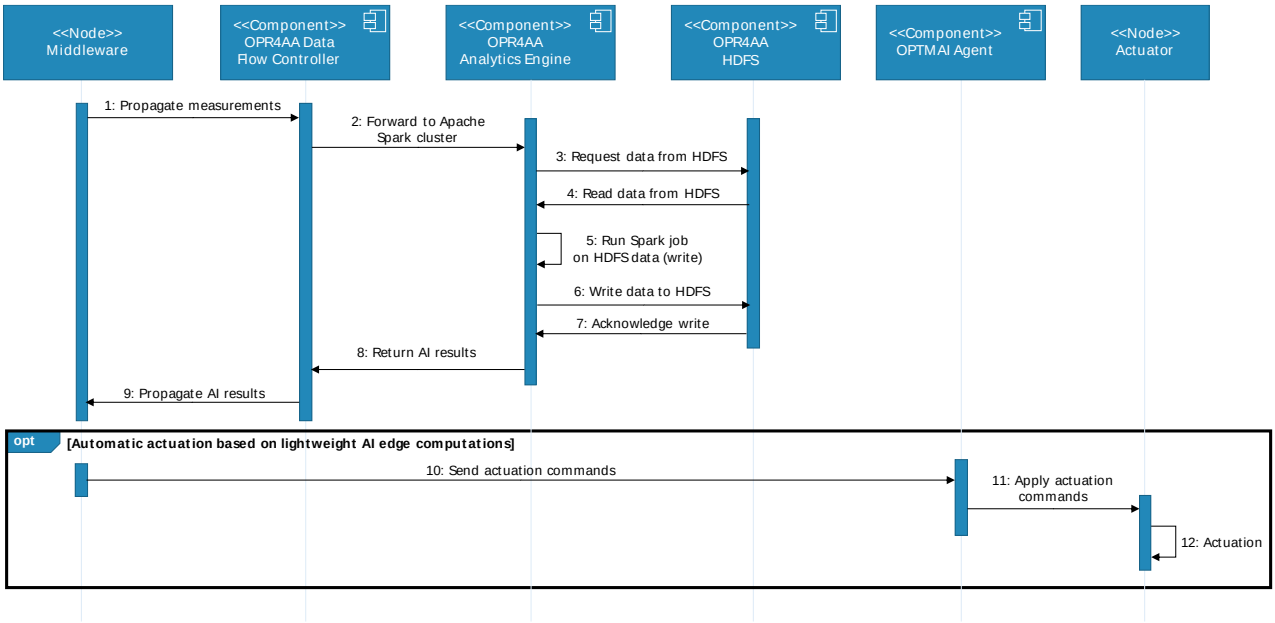


Figure 24: On-the-edge processing sequence diagram.

5.2.5 Manual and automatic re-configuration

The application scenario elaborated in the diagram shown in Figure 25 demonstrates how the Manufacturing (re-)configuration Service agents are triggered by the events registered at the Smart Quality Control algorithmic components (executing in parallel). After execution of the agent, re-configuration parameters can be translated to actuator commands that can be executed either automatically (through the OPTIMAI Agent), or manually, by presenting the option to apply re-configuration to the proper user at the proper terminal station. The user can opt to accept the re-configuration suggestion, in which case the Middleware triggers the OPTIMAI Agent after receiving clearance from the OMIDES Front-end/Back-end components. The application scenario is related to UC2, and, for the manual case, can be realized in UC3 as well.

An alternative diagram shown in demonstrates the alternative of suing the MAIAR software (through the wearable OPTIMAI smart glasses) to provide the operator with on-site decision support.

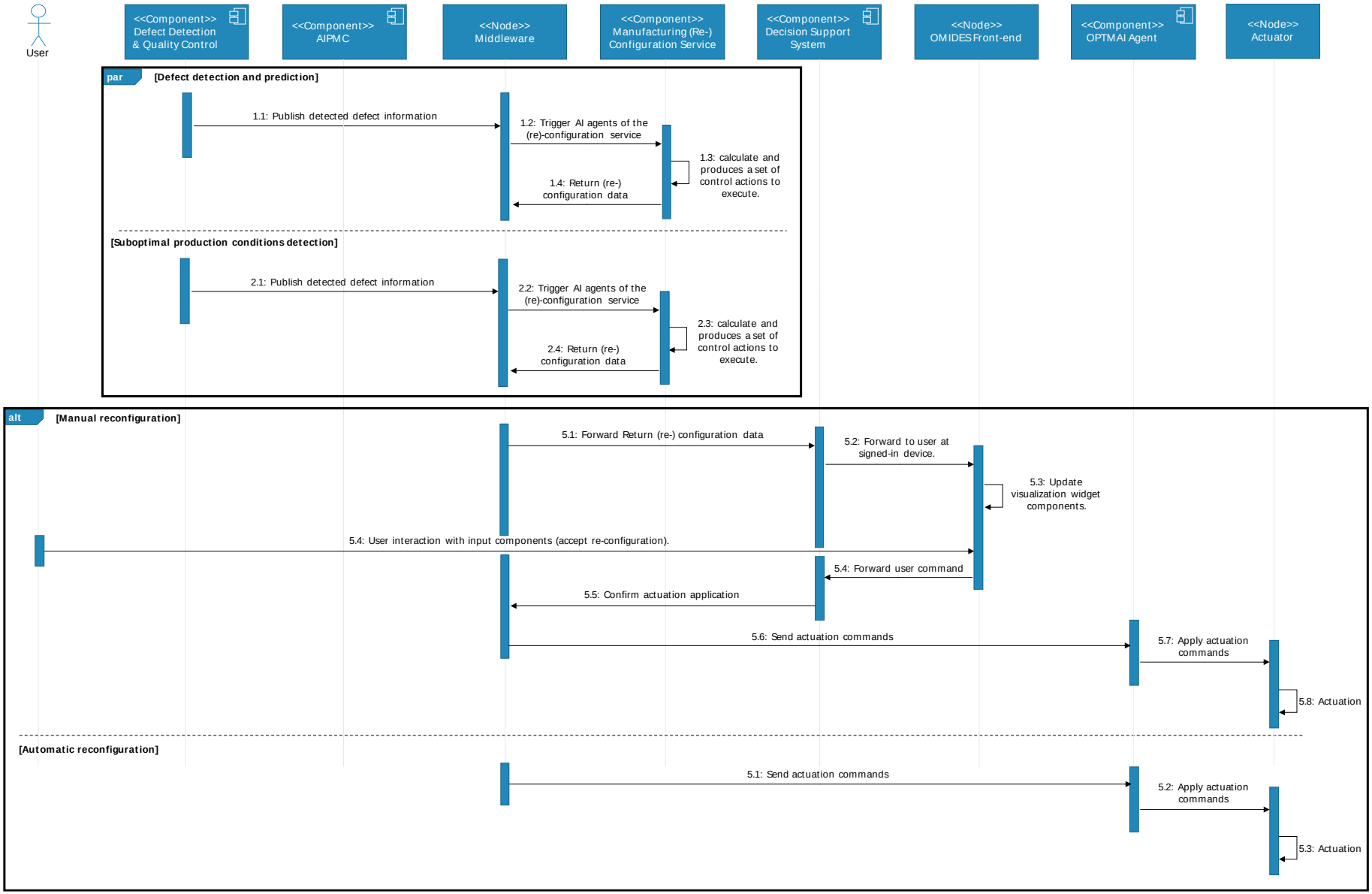


Figure 25: Manual and automatic reconfiguration sequence diagram (using OMIDES Front-end).

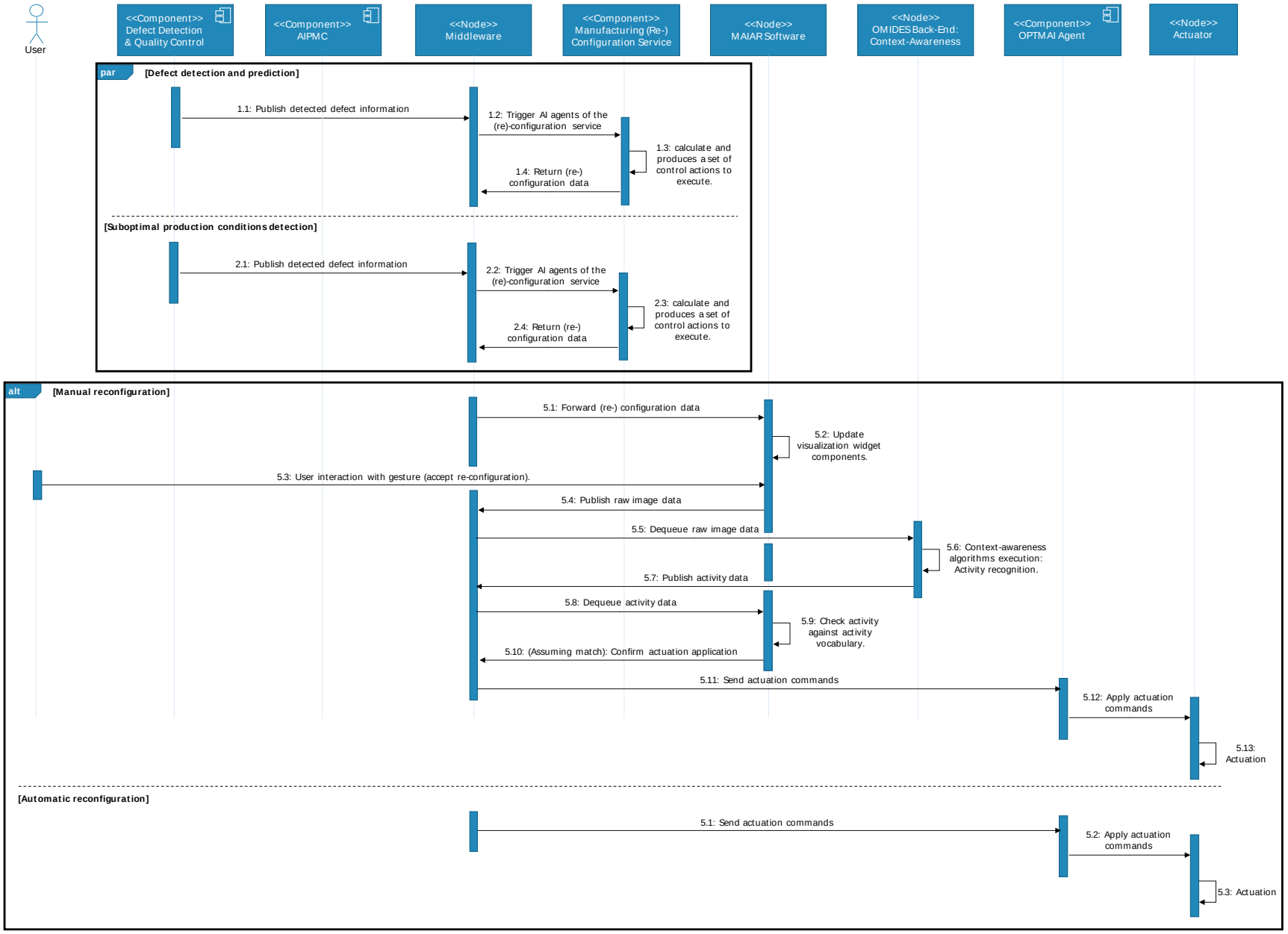


Figure 26: Manual and automatic reconfiguration sequence diagram (using MAIAR Software).

5.2.6 Software configuration transaction on the blockchain

The application scenario shown in the sequence diagram in Figure 27 elaborates on the means in which the Blockchain Framework components are triggered whenever a software configuration is to be applied on a sensor or actuator on the shop floor. This can for instance be executed whenever an operator elects to apply a new configuration based on the outputs of the AI Framework components. The data related to the configuration to be applied is forwarded by the Middleware to the Blockchain API Service. The latter forwards the data through the Web3.js endpoint to the Firmware/Software Validation smart contract, which in turn requests a new hash value for the transaction by the IPFS. After the has is obtained, the contract is executed on the EVM and the hash is stored on a new block.

A similar transaction is triggered in the firmware update case. The application scenario is related to UC2, and, for can be realized in UC3 (when a simulated condition reveals an optimal production environment, which the operator elects to apply in the long-term) as well.

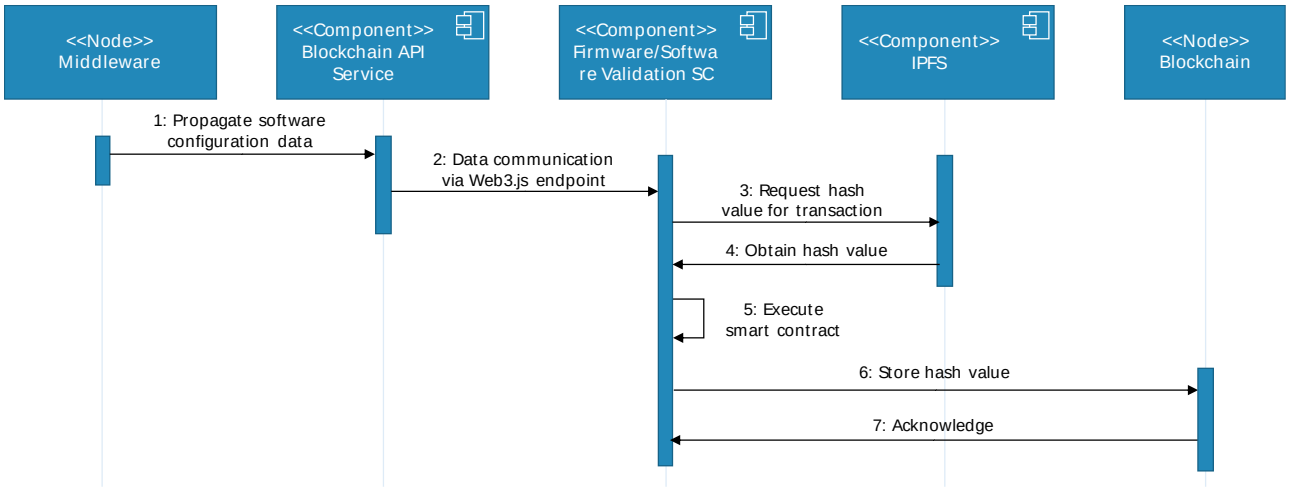


Figure 27: Software configuration transaction on the blockchain sequence diagram.

5.2.7 Production line simulation

In this scenario depicted in Figure 28, processes that are exclusively related to UC3 are elaborated. An operator decides to simulate a particular plant configuration using the simulation front-end client (Visual Components Software – VC4.4). The operator elects the process model and the Virtualized Sensor Network dataset (e.g., from a particular day) to load up. The information is obtained from the Historical Data silo in the Cloud Data Repository maintained by the Middleware. The Virtualized Sensor Network on boards the data and simulation can start at the behest of the user. The sensors data is generated as if measurements were collected from actual hardware, entering the Middleware in typical fashion. The Middleware handles cybersecurity at data acquisition and proceeds to forward the data to the architectural components by means of the endpoints API.

The system now enables different functions (as described in previous application scenarios), such as the display of higher level takeaways displayed in an OMIDES Front-end running terminal.

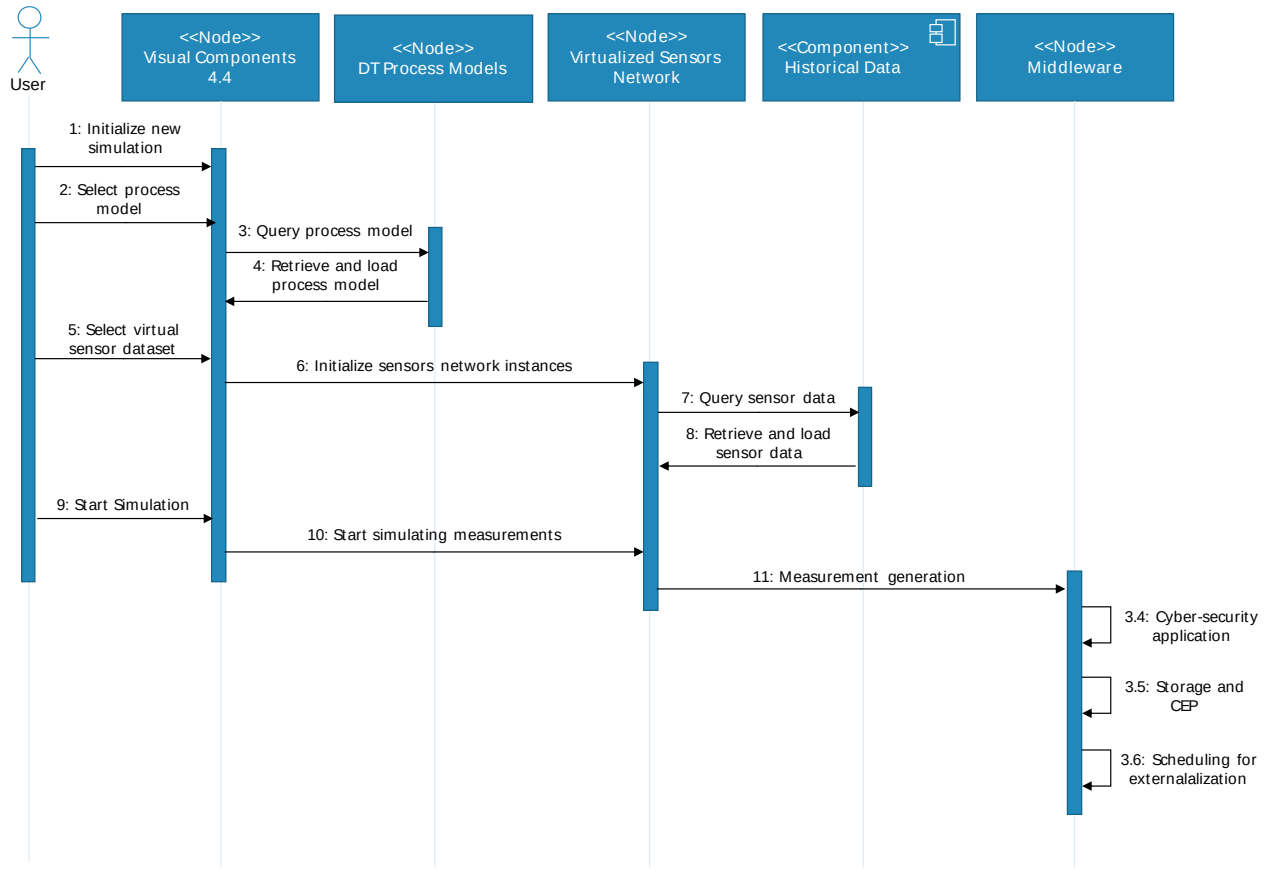


Figure 28: Production line simulation sequence diagram.

5.2.8 Intelligent Marketplace

This application scenario is demonstrated in Figure 29. As can be seen, whenever the Smart Quality Control components triggers a new defect event, the defective part information is automatically entered into the Intelligent Marketplace Back-End through the Back-end Service, and a new Scrap Reuse Offering is created.

A Product Line Operator (PLO) then logs into the Marketplace by invoking the proper log-in routines and being authenticated through the Marketplace IDM. The Dashboard reloads to the PLO dashboard View. The PLO can then see the automatic listing created, and further work on the scrap offering through functions supported by the Dashboard. Eventually, the Scrap Reuse Offering is published to third parties.

A Scrap Trader then logs into the Marketplace by similarly invoking the proper log-in routines and being authenticated through the Marketplace IDM. The Dashboard hence reloads to the Scrap Trader Dashboard View. The Scrap Trader can see the listing created for the Scrap Reuse Offering and elect to purchase the item.

The application scenario refers to zero-waste production, hence relates to UC1.

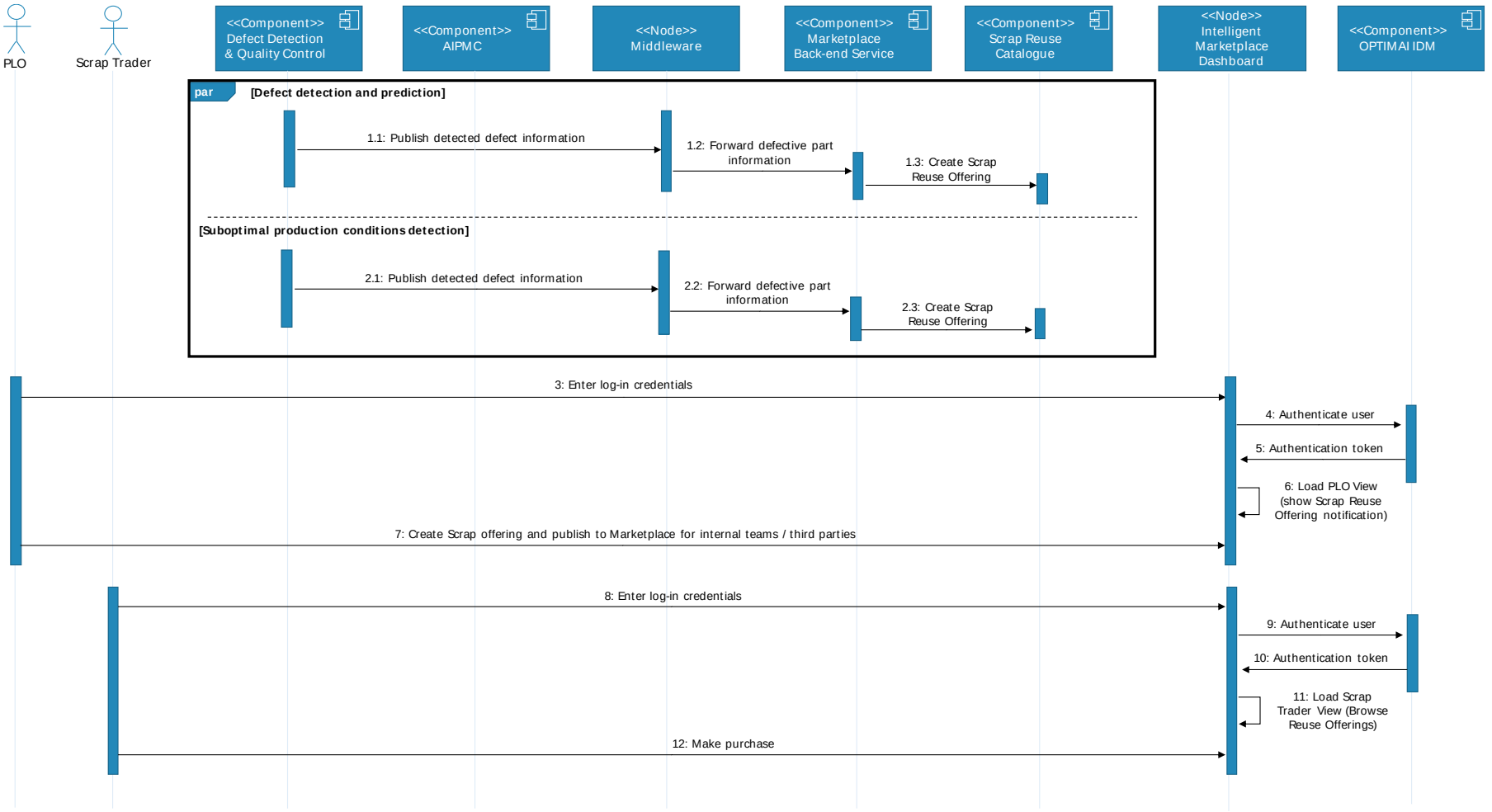


Figure 29: Intelligent marketplace sequence diagram

6 OPTIMAI Architecture - Deployment view

Typically, in software system architectures following the IEEE 1471 "Recommended Practice for Architectural Description for Software-Intensive Systems" [21], the 'Deployment' view specifies the physical deployment of the described system's components across physical and virtualised computing resources, together with their hardware and software requirements. Using this viewpoint, the system developers and integrators are in a better position to determine: (i) the number and specifications of the hardware necessary for the execution of particular system components; (ii) the location where these entities will be deployed; and (iii) means to map software blocks to hardware components. The means by which this information is conveyed is through a deployment diagram.

Section 6, D2.4, M12.

The deployment diagram for the OPTIMAI architectural stack is presented in Figure 30. A variety of devices will be utilized in the deployment, particularly with respect to the QCS Network, where different deployment considerations are available (hardware sensors, sensors connected to industrial PCs, sensors connected to computational modules for edge computing, etc.).

In addition, in this Section we present an updated topological view following the framework of the Industry 4.0 smart factory and its tangible layers, as specified in [22]. These are:

- *Physical resources. Constitutes the various kinds of "smart" physical resources, e.g., machinery, products and sensors/actuators, that make up the IoT network.*
- *Industrial network. Refers to the wireless network deployed for the support of both the intercommunication of physical resources as well as the connection of those resources to the Cloud.*
- *Cloud. Delivers Infrastructure-, Platform- and Software-as-a-Service (IaaS, PaaS, SaaS) solutions by virtualising both computing and storage capabilities that can scale in and out on demand. Within the context of OPTIMAI, this layer encompasses either private (i.e., on-premise/internal) or public (i.e., over the Internet) Cloud services, depending on strategic (e.g., the business already investing in on-premise data centres), or security concerns.*
- *Supervisory control terminals. Includes all end-user terminals (e.g., PCs, laptops, smart surfaces, head-mounted displays, etc.) that can access processing results and storage data housed within the Cloud, and display it to an end-user via UI/HMI solutions.*

For the purposes of OPTIMAI deployment, we proceed to extend this framework by introducing an Edge layer between the Physical resources and the Industrial network layers, reflecting the more recently introduced capacities afforded by the edge computing paradigm.

Section 6, D2.4, M12.

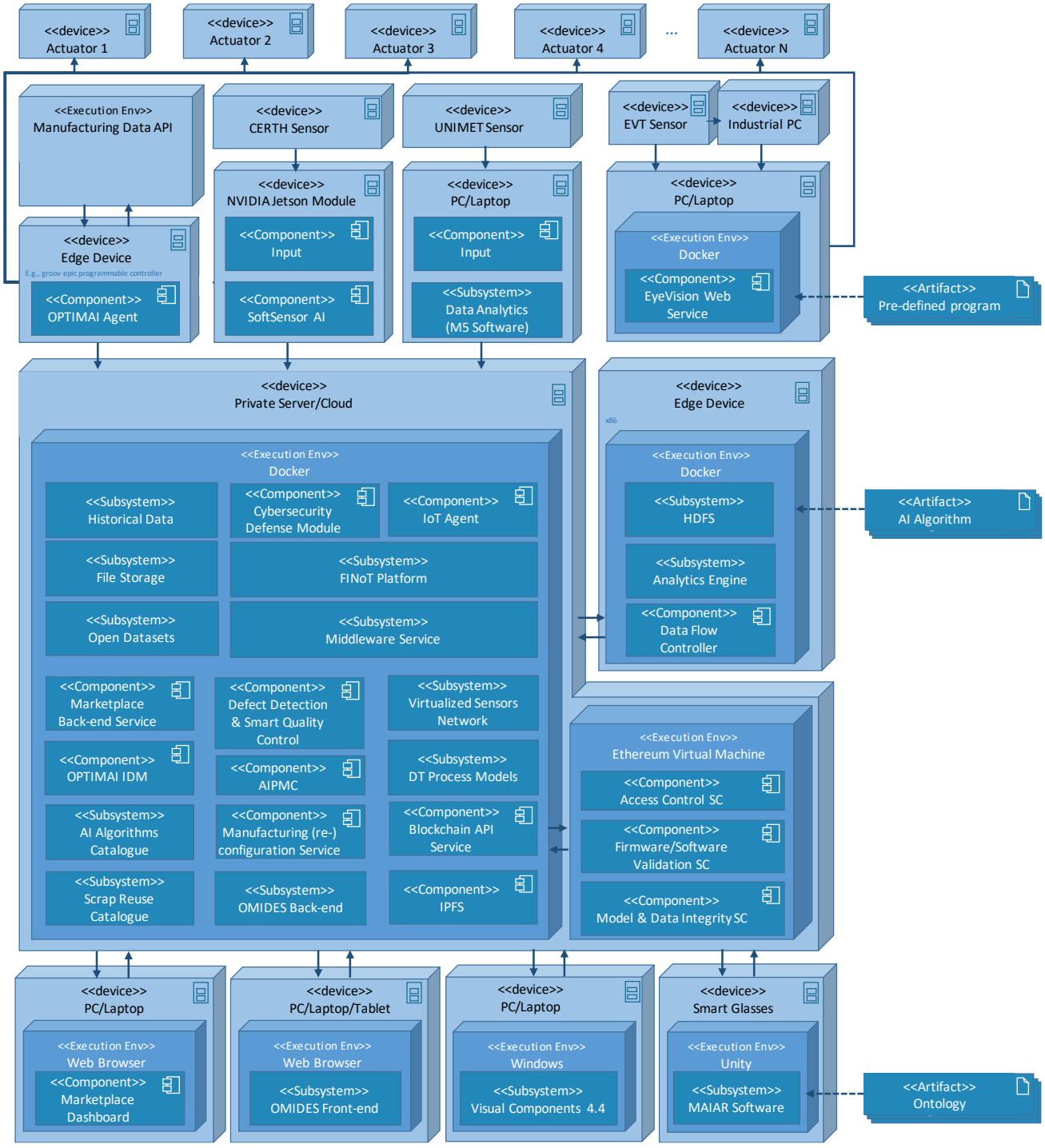


Figure 30: OPTIMAI Deployment diagram.

We present this updated topological diagram in Figure 31.

The depicted view organises components in the different smart factory framework layers, and their relation to the operational mechanism dual loop closed system [23]. The first loop considers elements that are involved in the coordination and feedback provided at the Cloud-level toward reconfiguring assets found in the Physical Resources Layer ("Coordinator"), while the second regards data visualisation and manipulation loop manifested between the Cloud

components engaged in statistical analysis ("Statistician"), and the supervisory terminal applications. Big data storage provisioned on the Cloud plays a central role in each loop, facilitating both sensing and actuation, as well as control manipulation processes in the smart factory framework.

Section 6, D2.4, M12.

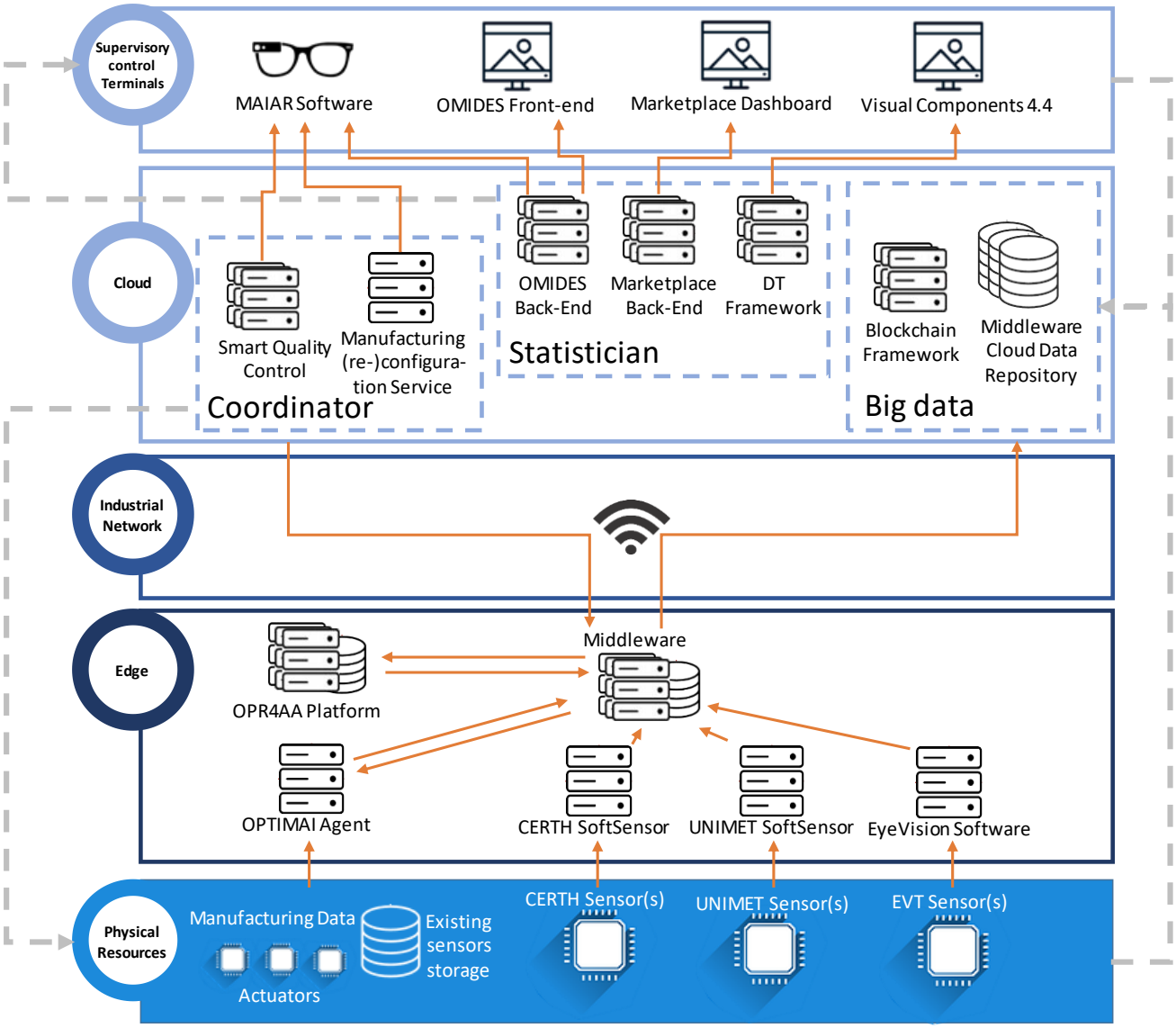


Figure 31: Topological view of OPTIMAI architecture through the smart factory framework perspective.

7 Conclusions

In this deliverable, the comprehensive overview of the final version of the OPTIMAI architecture has been presented, complete with the detailed specification of all of its functional elements, subsystems and main interaction rules and principles. The present document complements, and wherever noted, supersedes the contents of D2.4, and culminates the activities in Task 2.3: System specifications and architecture.

In this report we have elaborated on the architectural refinement process, which consisted of a revised cycle of technology exploration; top-down design and bottom-up refinement, following the provisions laid out in D2.4. In the first step, technology exploration, the progress made so far in the different tasks of the project was closely monitored, eliciting wherever necessary new components so as to accommodate the writings in the various project deliverables. On the top-down phase, the existing architecture was further refined and re-aligned, ultimately producing 38 (as opposed to the original 36) components and modules, organized into subsystems, topological categories and technological enablers addressed. During the final bottom-up process, partners were involved (using template information delivery) to further elaborate on the architectural subsystems where their components contribute, and the final architectural stack was finalized over two architectural workshops.

The REPORT further specifies foreseen information flows for indicative scenarios in each of the three use case types defined for the OPTIMAI project. These flows are defined by means of sequence diagrams, aiming at clearly demonstrating *"how the provisioned services will communicate information within the confines of the use case"* (D2.4).

Finally, the foreseen system deployment has been fully elaborated to enable the considerations of specific provisions for the execution of the use cases. A complete deployment diagram is presented, alongside the final mapping of the architectural stack to the smart factory framework architectural layers [22].

An important aspect throughout this process has been to maintain and update the alignment of the architecture to RAMI 4.0, which demonstrates pragmatically how OPTIMAI upholds specific principles and guidelines regarding smart factory implementation, along with underlining useful integration aspects for Work Packages 6 and 7. This is an important outcome of Task 2.3, further highlighted by the recently accepted publication regarding the OPTIMAI architectural mapping to RAMI 4.0 (and the IIRA), accepted for presentation at the 27th IEEE Annual Conference on Emerging Technologies and Factory Automation (ETFA) [24].

References

[1] DIN, DKE. (2020). German Standardization Roadmap Industrie 4.0 Version 4.0. *DIN and DKE Roadmap*. URL: https://www.din.de/en/innovation-and-research/industry-4-0/german-standardization-roadmap-on-industry-4-0-77392

[2] Monostori, L. (2014). Cyber-physical production systems: Roots, expectations and R&D challenges. *Procedia CIRP, 17*, 9-13.

[3] Nakagawa, E. Y., Antonino, P. O., Schnicke, F., Capilla, R., Kuhn, T., & Liggesmeyer, P. (2021). Industry 4.0 reference architectures: State of the art and future trends. *Computers & Industrial Engineering, 156*, 107241.

[4] Plattform Industrie 4.0 (2018). *Reference Architectural Model Industrie 4.0 (RAMI4.0) - An Introduction*. URL: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.html

[5] Pisching, M. A., Pessoa, M. A., Junqueira, F., dos Santos Filho, D. J., & Miyagi, P. E. (2018). An architecture based on RAMI 4.0 to discover equipment to process operations required by products. *Computers & Industrial Engineering, 125*, 574-591.

[6] Baptista, L. F., & Barata, J. (2021). Piloting Industry 4.0 in SMEs with RAMI 4.0: an enterprise architecture approach. *Procedia Computer Science, 192*, 2826-2835.

[7] Industrial Internet Consortium. (2019). *The Industrial Internet of Things Volume G1: Reference Architecture. Version 1.9 June 19, 2019*. URL: https://www.iiconsortium.org/pdf/IIRA-v1.9.pdf.

[8] Lin, S-W, Murphy, B., Clauer, E., Loewen, U., Neubert, R., Bachmann, G., Pai, M., & Hankel, M. (2017). *Architecture Alignment and Interoperability: An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper* [White paper]. Industrial Internet Consortium. URL: https://www.iiconsortium.org/pdf/JTG2_Whitepaper_final_20171205.pdf

[9] Industrial Value Chain Initiative. (2016). *Industrial Value Chain Reference Architecture (IVRA)*. URL: https://docs.iv-i.org/doc_161208_Industrial_Value_Chain_Reference_Architecture.pdf

[10] IBM (2021). Industry 4.0 architecture for manufacturing. IBM Cloud Architecture Center. URL: https://www.ibm.com/cloud/architecture/files/industry-40-architecture-pdf-template.pdf

[11] Kassner, L., Gröger, C., Königsberger, J., Hoos, E., Kiefer, C., Weber, C., Silcher, S., & Mitschang, B. (2016). The Stuttgart IT architecture for manufacturing. In *International Conference on Enterprise Information Systems* (pp. 53-80). Springer, Cham.

[12] Resman, M., Pipan, M., Šimic, M., & Herakovič, N. (2019). A new architecture model for smart manufacturing: A performance analysis and comparison with the RAMI 4.0 reference model. *Advances in Production Engineering & Management, 14*(2), 153-165.

[13] Dainow, B., & Brey, P. (2021). *Ethics By Design and Ethics of Use Approaches for Artificial Intelligence*. Retrieved from https://ec.europa.eu/info/funding-tenders/opportunities/docs/2021-2027/horizon/guidance/ethics-by-design-and-ethics-of-use-approaches-for-artificial-intelligence_he_en.pdf

[14] Hashmi, M., Casanovas, P., & de Koker, L. (2018). Legal Compliance Through Design: Preliminary Results. In *Proceedings of the 2nd Workshop on Technologies for Regulatory Compliance* (pp. 59–72).

[15] Javaid, M., Haleem, A., Singh, R. P., Khan, S., & Suman, R. (2021). Blockchain technology applications for Industry 4.0: A literature-based review. *Blockchain: Research and Applications*, 100027.

[16] Park, H. S., & Febriani, R. A. (2019). Modelling a Platform for Smart Manufacturing System. *Procedia Manufacturing, 38*, 1660-1667.

[17] Elkhawas, A. I., & Azer, M. A. (2018, December). Security perspective in rami 4.0. In *2018 13th International Conference on Computer Engineering and Systems (ICCES)* (pp. 151-156). IEEE.

[18] Hossain, M. T., Badsha, S., & Shen, H. (2020, September). PoRCH: A novel consensus mechanism for blockchain-enabled future SCADA systems in smart grids and industry 4.0. In *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)* (pp. 1-7). IEEE.

[19] Steindl, G., Stagl, M., Kasper, L., Kastner, W., & Hofmann, R. (2020). Generic digital twin architecture for industrial energy systems. *Applied Sciences, 10*(24), 8903.

[20] Jeffay, K. (1993). The real-time producer/consumer paradigm: A paradigm for the construction of efficient, predictable real-time systems. In *Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice* (pp. 796-804).

[21] IEEE. (2000). IEEE Recommended Practice for Architectural Description for Software-Intensive Systems (1471-2000). IEEE. URL: https://ieeexplore.ieee.org/document/875998

[22] Wang, S., Wan, J., Li, D., & Zhang, C. (2016). Implementing smart factory of industrie 4.0: an outlook. *International journal of distributed sensor networks, 12*(1), 3159805.

[23] Wang, S., Wan, J., Zhang, D., Li, D., & Zhang, C. (2016). Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. *Computer networks, 101*, 158-168.

[24] Margetis, G., Apostolakis, K. C., Dimitriou, N., Tzovaras, D., & Stephanidis, C. (2022, September). Aligning Emerging Technologies onto I4.0 principles: Towards a Novel Architecture for Zero-defect Manufacturing. In *2022 27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (to appear). IEEE.

Appendix A – Component refinement template

Figure 32: Architecture refinement – Component template distributed to partners (Cover page).

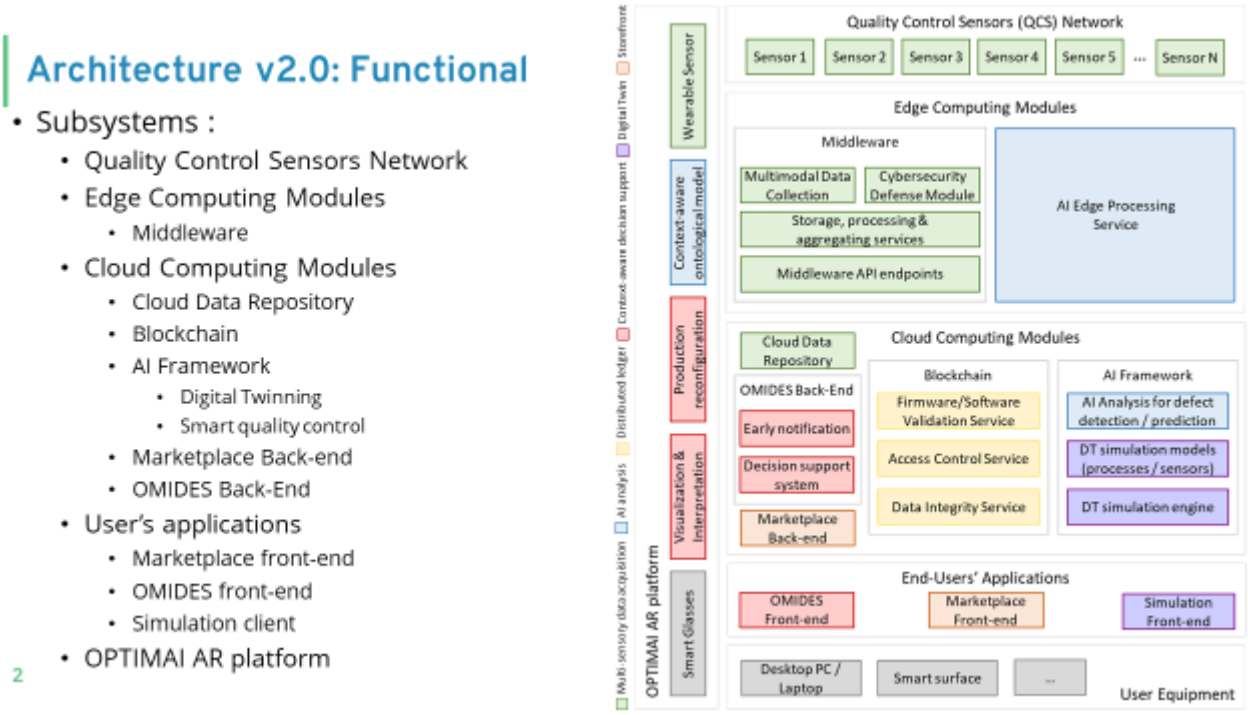


Figure 33: Architecture refinement – Component template distributed to partners (page 2).

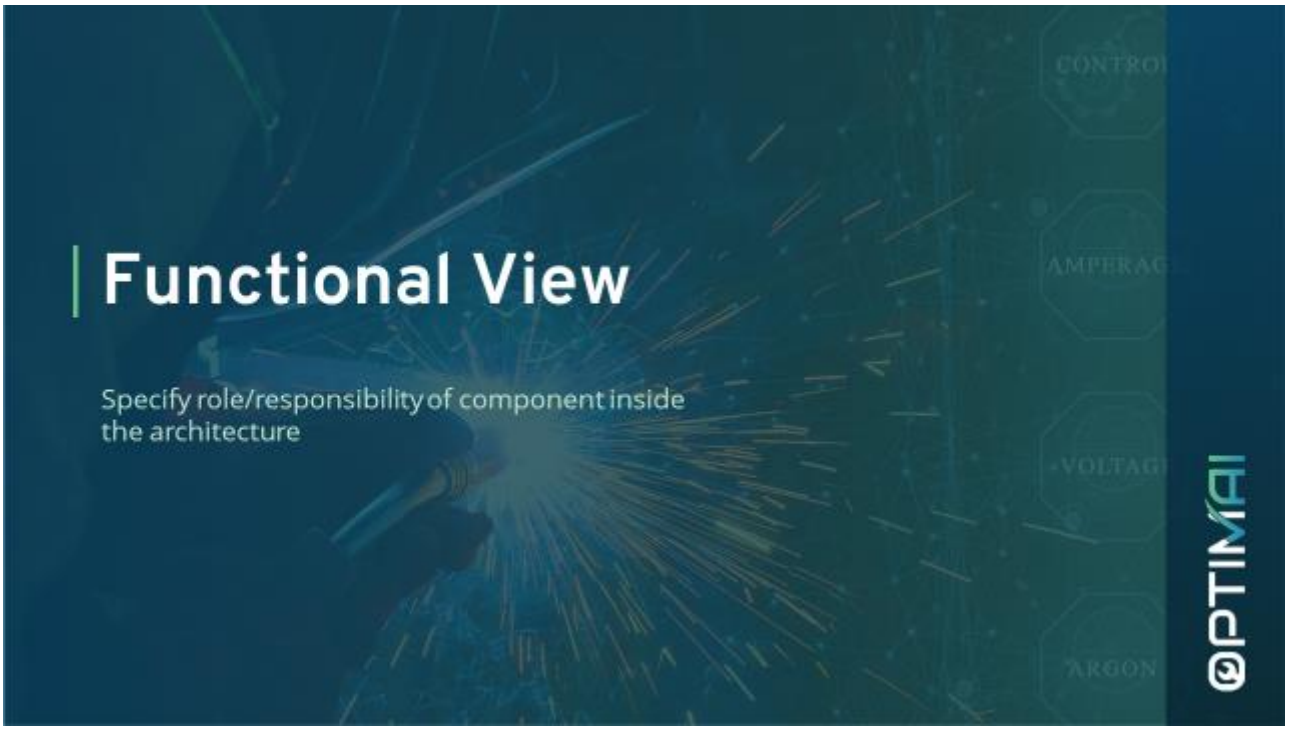


Figure 34: Architecture refinement – Component template distributed to partners (page 3).

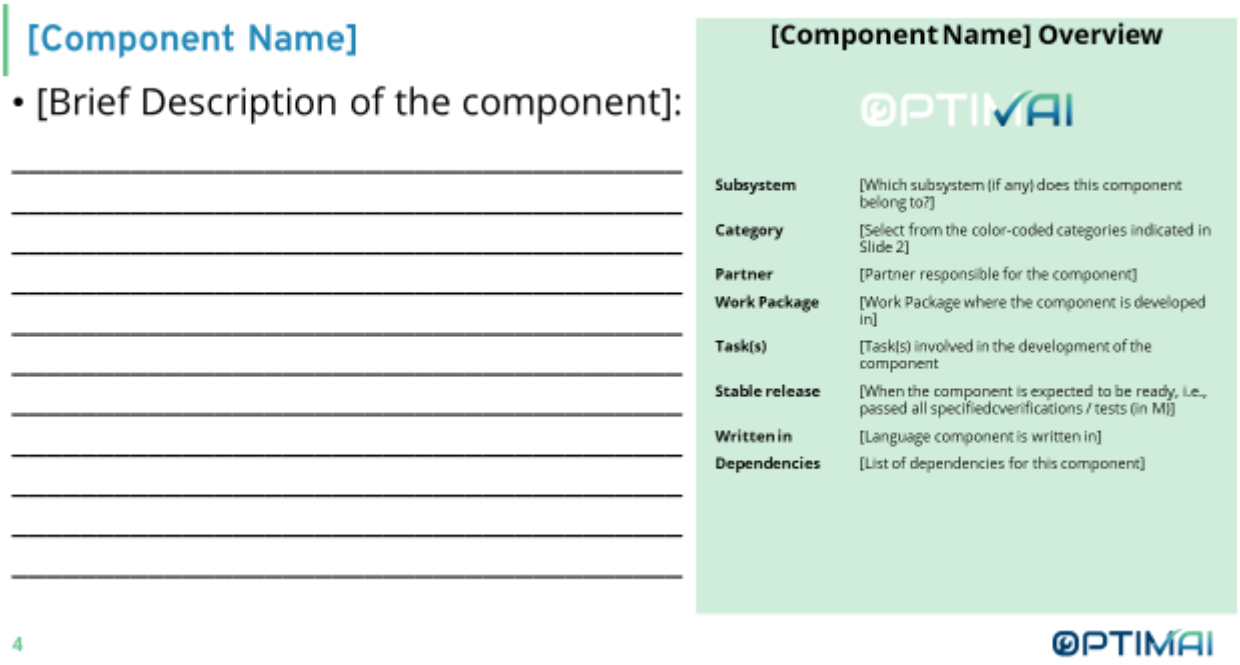


Figure 35: Architecture refinement – Component template distributed to partners (page 4).

[Component Name] Functional aspects

- List of Key Features
  - [Bullet list of the main features provisioned by this component, *e.g.*, *"authenticates the user by means of a set of credentials including a username and a password"*...]
  - ...

5

OPTIMAI

Figure 36: Architecture refinement – Component template distributed to partners (page 5).

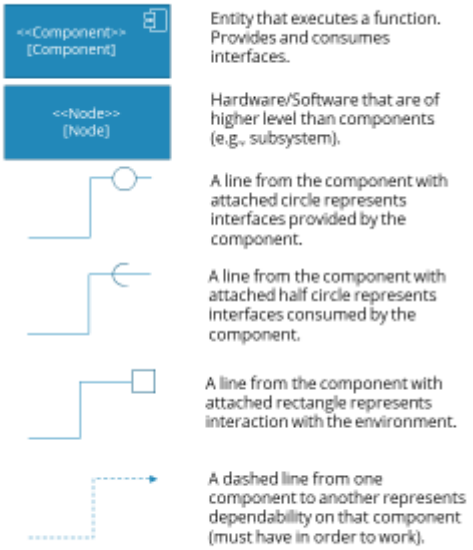
[Component Name] Internal architecture

Component Diagram

[Copy and paste the UML component diagram shapes and symbols shown to the right (In Groups, editable), to build a comprehensive component diagram that represents the a bird's-eye view of the software].

[Resize and edit as you see fit. Replace the words in [brackets] only].

[Delete this text and extra shapes when you are done].



6

OPTIMAI

Figure 37: Architecture refinement – Component template distributed to partners (page 6).

Figure 38: Architecture refinement – Component template distributed to partners (page 7).

[Component Name] provided Interfaces

This section defines the interfaces exposed by the [Component]

- [**Interface Name**]: [Brief description of the functions supported by this interface] **Consumed by:** [Other Component].
- … (list all)

This interface implements a [synchronous/asynchronous] communication mode

8 OPTIMAI

Figure 39: Architecture refinement – Component template distributed to partners (page 8).

Select the appropriate set of slides for synchronous (Slides 10-11) and Asynchronous (Slides 12-13) communication and fill in the information in brackets ([...]).

Follow-up each interface with the request (Slide 14) and (if relevant) response (Slide 15) parameters

Each interface should amount to a total of 3-4 slides.

Duplicate as necessary for all interfaces.

9

Figure 40: Architecture refinement – Component template distributed to partners (page 9).

[Interface #1] information elements

(For synchronous communication, e.g., request-response schema)

[Interface #1] allows the following operations:

- [Component] (#1) sends [Other Component] (#2) a request to …
- [Other Component] responds with …

Operation (message)	Requirement	Direction	Message standard	Message format	Method
e.g., *Request*	e.g., *Mandatory, Optional*, etc.	#1 → #2	e.g., *REST, UDP*, etc.	e.g., *JSON, YAML*, etc.	e.g., *GET, POST*, etc.
e.g., *Response*	e.g., *Mandatory, Optional*, etc.	#2 → #1	e.g., *REST, UDP*, etc.	e.g., *JSON, YAML*, etc.	e.g., *GET, POST*, etc.

10

Figure 41: Architecture refinement – Component template distributed to partners (page 10).

[Interface #1] information elements

(For synchronous communication, e.g., request-response schema)

Endpoints:

Operation (message)	
e.g., *GET*	[endpoint]
e.g., *POST*	[endpoint]

Figure 42: Architecture refinement – Component template distributed to partners (page 11).

[Interface #1] information elements

(For asynchronous communication, e.g., publish/subscribe mechanism)

[Interface #1] allows the following operations:

- [Component] (#1) dispatches message with …
- [Other Component(s)] consume(s) message by subscribing to the proper exchange/topic/queue with …

Operation (message)	Broker	Message standard	Message format	Exchange Type
e.g., *Publish*	e.g., *RabbitMQ*, etc.	e.g., *AMQP*, *MQTT*, etc.	e.g., *JSON*, *YAML*, etc.	e.g., *Direct*, *Fanout*, *Topic*, *Headers*

Figure 43: Architecture refinement – Component template distributed to partners (page 12).

[Interface #1] information elements

(For asynchronous communication, e.g., publish/subscribe mechanism)

Endpoints:

Operation (message)	
e.g., *Queue*	[endpoint]
e.g., *Exchange*	[endpoint]

13

Figure 44: Architecture refinement – Component template distributed to partners (page 13).

[Interface #1] request parameters

The parameters sent when [sending a request to Other Component] / publishing a message] for the [name of the operation] operation will generally follow the indications below:

Parameter	Requisitness	Cardinality	Type	Description
Parameter name	*M* = Mandatory *O* = Optional *CM* = Conditional mandatory *CO* = Conditional optional	Number of allowed appearances of this parameter in message payload. Could be a range. Optional params could have cardinality *0*.	Type of data communicated, e.g., *string*, *integer*, etc.	Short description. Explain conditionality if relevant.

14

Figure 45: Architecture refinement – Component template distributed to partners (page 14).

[Interface #1] response parameters

The parameters returned by [Other Component] for the [name of the operation] operation will generally follow the indications below:

Parameter	Requisitness	Cardinality	Type	Description
Parameter name	*M* = Mandatory *O* = Optional *CM* = Conditional mandatory *CO* = Conditional optional	Number of allowed appearances of this parameter in message payload. Could be a range. Optional params could have cardinality *0*.	Type of data communicated, e.g., *string*, *integer*, etc.	Short description. Explain conditionality if relevant.

15

OPTIMAI

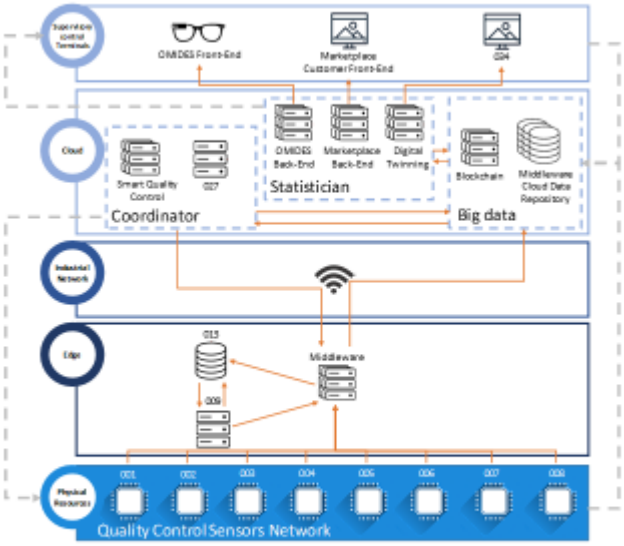
Figure 46: Architecture refinement – Component template distributed to partners (page 15).

Figure 47: Architecture refinement – Component template distributed to partners (page 16).

Topological placement

Within the Industry 4.0 smart factory framework tangible layers, [Component] is assigned to [select one]:

- **Physical resources** ("smart physical resources, e.g. machinery, products, sensors, actuators").
- **Edge** (supports edge computing capacities)
- **Industrial network** (wireless network deployed for intercommunication and connection to the cloud).
- **Cloud** (Public/Private IaaS, PaaS, SaaS, virtualising computing and storage capabilities)
- **Supervisory control terminals** (end-user terminals that access the processing results and display UI solutions).



Version 1.0, retrieved from D2.4

17

Figure 48: Architecture refinement – Component template distributed to partners (page 17).

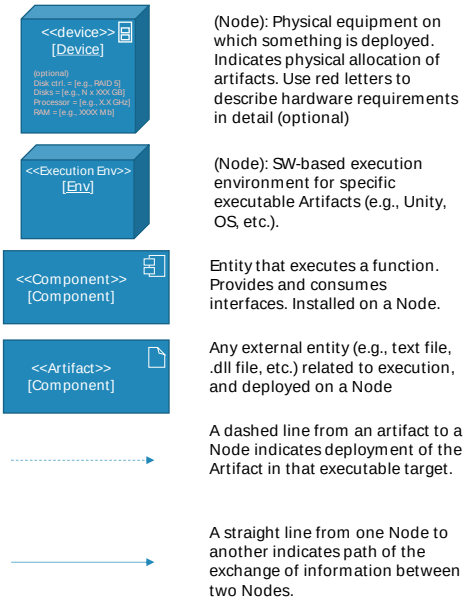
[Component Name] deployment environment

Deployment Diagram

[Copy and paste the UML deployment diagram shapes and symbols shown to the right (In Groups, editable), to build a comprehensive deployment diagram that represents hardware and software necessary for execution of the component functions].

[Resize and edit as you see fit. Replace the words in [brackets] only].

[Delete this text and extra shapes when you are done].



(Node): Physical equipment on which something is deployed. Indicates physical allocation of artifacts. Use red letters to describe hardware requirements in detail (optional)

(Node): SW-based execution environment for specific executable Artifacts (e.g., Unity, OS, etc.).

Entity that executes a function. Provides and consumes interfaces. Installed on a Node.

Any external entity (e.g., text file, .dll file, etc.) related to execution, and deployed on a Node

A dashed line from an artifact to a Node indicates deployment of the Artifact in that executable target.

A straight line from one Node to another indicates path of the exchange of information between two Nodes.

18

Figure 49: Architecture refinement – Component template distributed to partners (page 18).

Figure 50: Architecture refinement – Component template distributed to partners (back cover).