Integration Plan and Definition of Interfaces

CoRoSect.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016953

Author(s)/Organisation(s)	Hochschule Emden-Leer [HSEL]
Contributor(s)	HSEL, OAMK, Robotnik, AGVR, ATOS, UM and CERTH
Work Package	WP9. Secure platform integration
Delivery Date (DoA)	31.12.2022
Actual Delivery Date	06.02.2023
Abstract:	The document presents the first version of the Integration plan and definition of interfaces, the document details regarding the approach of integration of Shop floor components/assets by providing a standard interface for each asset to expose its functionalities according to Industry 4.0 specifications and standards. Asset administration shells offer effortless data integration, provide comprehensive semantic data homogenization, and convey contextual information to achieve the objectives of CoRoSect.

Document Revision History			
Date	**Version**	**Author/Contributor/ Reviewer**	**Summary of main changes**
28/10/2022	0.1		Table of Contents
08/12/2022	1.0		First draft
21/12/2022	1.2		First draft with HSEL internal revisions
08/01/2023			Final version

Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the EC Services)	
RE	Restricted to a group specified by the consortium (including the EC Services)	
CO	Confidential, only for members of the consortium (including the EC)	

Funding Scheme: Innovation Action (IA) ● Topic: H2020-ICT-46-2020

Start date of project: 01 January, 2021 ● Duration: 36 months

CoRoSect Consortium			
Participant Number	**Participant organisation name**	**Short**	**Country**
1	UNIVERSITEIT MAASTRICHT	UM	NL
2	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS	CERTH	GR
3	HOCHSCHULE EMDEN/LEER	HSEL	GER
4	LUONNONVARAKESKUS	LUKE	FIN
5	OULUN AMMATTIKORKEAKOULU OY - OULU UNIVERSITY OF APPLIED	OAMK	FIN
6	FUNDACION PARA LAS TECNOLOGIAS AUXILIARES DE LA AGRICULTURA	TECNOVA	ES
7	KATHOLIEKE UNIVERSITEIT LEUVEN	KU LEUVEN	BEL
8	HSEL IT SOLUTIONS AND SERVICES IBERIA SL		ES
9	ROBOTNIK AUTOMATION SLL	ROB	ES
10	AGVR BV	AGVR	NL
11	NASEKOMO AD	NASEKOMO	BG
12	ENTOMOTECH SL	ENTOMOTECH	ES
13	ENTOCYCLE LTD	ENTOCYCLE	GB
14	SOCIETA AGRICOLA ITALIAN CRICKET FARM SRL	ICF	IT
15	INVERTAPRO AS	INVERTAPRO	NOR
16	FIELD LAB ROBOTICS BV	FLR	NL
17	Food Scale Hub	FSH	RS
18	Agrifood Lithuania DIH	AFL	LT
19	CENTRO INTERNAZIONALE DI ALTISTUDI AGRONOMICI MEDITERRANEI	CIHEAM	IT

Table of Contents

Executive Summary

The deliverable details the approach towards integrating all shop floor components in a coherent system of cyber physical components using **standardized interfaces** as envisaged by the Industry 4.0 consortiums. The interfaces use the **Industry 4.0 recommended AAS** as a building block for implementation. An asset administration shell (AAS), as defined in the context of the Reference Architectural Model for Industry 4.0 (RAMI 4.0), is a practical embodiment of the latest buzzword, **digital twin**, and can be realized with the integration of operation technologies and information and communication technologies. AASs offer an interoperable way to capture key information pertaining to assets, such as intrinsic properties, operational parameters, and technical functionalities, and to enable straightforward interaction over standardized, secure communication with other Industry 4.0 components. The goal here is to present the status quo of AAS development **for the shop floor components** in the different CoRoSect Application Scenarios, to design an intuitive method for implementing AASs, and to develop an AAS-enabled digital solution for cyber-physical applications in the insect rearing industry that is addressed in the project.

For the development of AASs, *AAS package* explorer will be used. The contents of the AAS will be based on the standards specified in the literatures of "Platform Industry 4.0" reports and publications. Industry 4.0 compliant communication technologies will be used in implementing the interfaces from Operation Technology (OT) level to the Information Technology (IT) level.

These defined interfaces (AAS) will expose the set of properties (datasets) and operations (commands) supported by each CoRoSect's component and provide the key link for the I4.0 compliant integrated CoRoSect platform (D9.2).

List of Figures

List of Abbreviations and Acronyms	
AAS	Asset Administration Shell
API	Application Program Interface
AGV	Autonomous Guided Vehicle
DIN SPEC	Deutsches Institut für Normung Specification
DoA	Document of Action
DoW	Document of Work
D-Robot	Stacking - De-stacking Robot
DSS	Decision Support System
HSEL	Hochschule Emden Leer
I4.0	Industry 4.0
ICPS	Industrial Cyber-Physical System
I-Crate	Intelligent Crate
IEC	International Electrotechnical Commission
IIoT	Industrial IoT
IMS	Information Management System
IoT	Internet of Things
ISA	International Society of Automation
IT	Information Technologies
JSON	JavaScript Object Notation
M12	Month 12
MES	Manufacturing Execution System
MQTT	Message Queue Telemetry Transport
M-Robot	Manipulation-Robot
NGSI	Next Generation Systems Interfaces
OAuth	Open Authorization
OPC-UA	OLE (Object Linking and Embedding) for Process Control-Unified Architecture
OT	Operational Technology levels
RAMI4.0	Reference Architecture Model for I4.0
REST API	Representational state transfer Application Program Interface
ROS	Robot Operating System
SC	Service Consumer
SV	Service Provider
SFM	Shop Floor Manager
SoA	Service oriented Architecture
SQL	Structured Query Language
XML	Extendible Markup Language

1. Introduction

1.1. Scope and objectives of the deliverable

According to CoRoSect's DoA, the project "*will bring new insight to automated insect farming by introducing a novel digitalized integrated robotic solution based on the Reference Architecture Model Industry 4.0 (RAMI4.0) implemented as an Industrial Cyber-Physical System (ICPS) to be able to support all phases of the insects' lifecycle inside insect farms. The fundamental aim of the system (and the great innovation it provides) will be to provide repetitive but also cognitively and physically demanding tasks, like transferring and handling of crates (de-stacking and stacking), monitoring of environmental conditions, larvae separation/detection, insect feeding, which require increased manual effort or continuous human supervision, with correspondingly automatic robotic-based procedures, as service in an I40-compliant Information-Communication Infrastructure*".

The main objective of this deliverable is to **present the Integration Plan and define the software interfaces** that support the main objective addressed here above.

The task also recommends the use of safety standards to implement a "*collaboration environment, where humans and robots will harmoniously share and undertake at the same time different processing and manipulation tasks, targeting the application case of insect farming*" on the basis of requirement provided by end-users, one of the core objectives of this implementation, is to create a safe working environment where the humans work in association with robots to navigate and circumvent the existing hazardous environments in the insect farming world.

The current industrial scenario demands that the manufacturing world exploits the technology available in the IIoT infrastructure (e.g., Smart systems) and communication frameworks to perform critical tasks automatically and with the ability to self-monitor and self-correct. This deliverable analyses the **interfaces from the shop floor component providers**. Moreover, it **provides an IIoT reference framework** that can be implemented in different use cases.

The **RAMI 4.0** [1] serves as a framework for the implementation of interfaces and integration of assets and systems applying the service-oriented architecture paradigm. The 6 layers of the RAMI 4.0 digitalization dimension are referenced in this implementation along with the value stream and life cycle dimension – IEC 62890 [2]. The key aspect is to integrate the systems from the shop floor (OT) with the systems from the IT level, such as e.g., ERP, MES, etc.) using standard communication protocols and frameworks. The integration of OT and IT requires, among other features, portability, connectivity and interoperability.

The Asset Administration Shell (AAS) of each asset from the insect production environment is recommended as digitalization and integration technology by the DIN SPEC 91345 [1] and the guidelines described in [3], i.e., turning assets into I4.0 components.

1.2. Relationships with other deliverables and tasks

The integration plan proposed by this deliverable will act as a pipeline for data flow from assets between OT and IT levels of the hierarchy level of ISA 95, when it is mapped into the information-control-automation infrastructure of the CoRoSect environment. Work packages in this project use asset data for several functionalities such as orchestration, safety, security, process-analysis and -planning, function-testing, etc.

The relationship of WP9 with other Work packages is detailed below and also in Figure 1.

1. WP2 (Use-cases, user requirements and system architectures): Integration plan requires a defined system architecture proposed in WP2 and this work package is closely interlinked with WP9 Task 9.1, Task 2.3 (System architecture) [4] requires the proposed interfaces and vice versa. As the CoRoSect architecture will need both the Robots and Humans working in a harmonious manner, the Service oriented Information management system needs to ensure both the software and hardware assets stay updated and co-ordinated.
2. WP4 (Farm-level modelling and orchestration): requires the orchestration of functions of the CoRoSect system, the MES requires the integration of data generated by the Cyber physical assets to optimize workflows in farms, the service-oriented Information management system will work based on the integration of asset data using standard industrial protocols.
3. WP5 deals with vision tasks related to perceiving the surrounding work environment in the farms, to this end the task involves defining the interfaces for object detection and VR headset (Augmented Reality glasses (Hololens)).
4. WP6 (Robotic actions planning and control) As humans and Robots coexist in the CoRoSect ecosystem, the safety aspect is vital, Task 6.4 requires data from Robots, I-Crates, status of the proposed and current tasks to optimize the plan and route the robots will take. The implemented controllers on these assets will expose services and receive information from MES.
5. WP7 (Cognitive robots and smart mechatronics) requires the Robot Cell communicating with other systems from other work packages. This WP tasks require integration of data that can be used as information for manipulation of robots based on information received by their corresponding controllers and sensors (ICRATEs) that are implemented to integrate the CoRoSect System.
6. WP8 deals with AGV route planning, a mechanism for safe Human Robot collaborations. So, this deliverable provides interfaces that are needed to implement the tasks in WP8.
7. WP10 requires testing and evaluation of CoRoSect system interfaces instance at Insect farms across Europe. This step goes a long way into implementing the interfaces developed in real-world like scenarios.

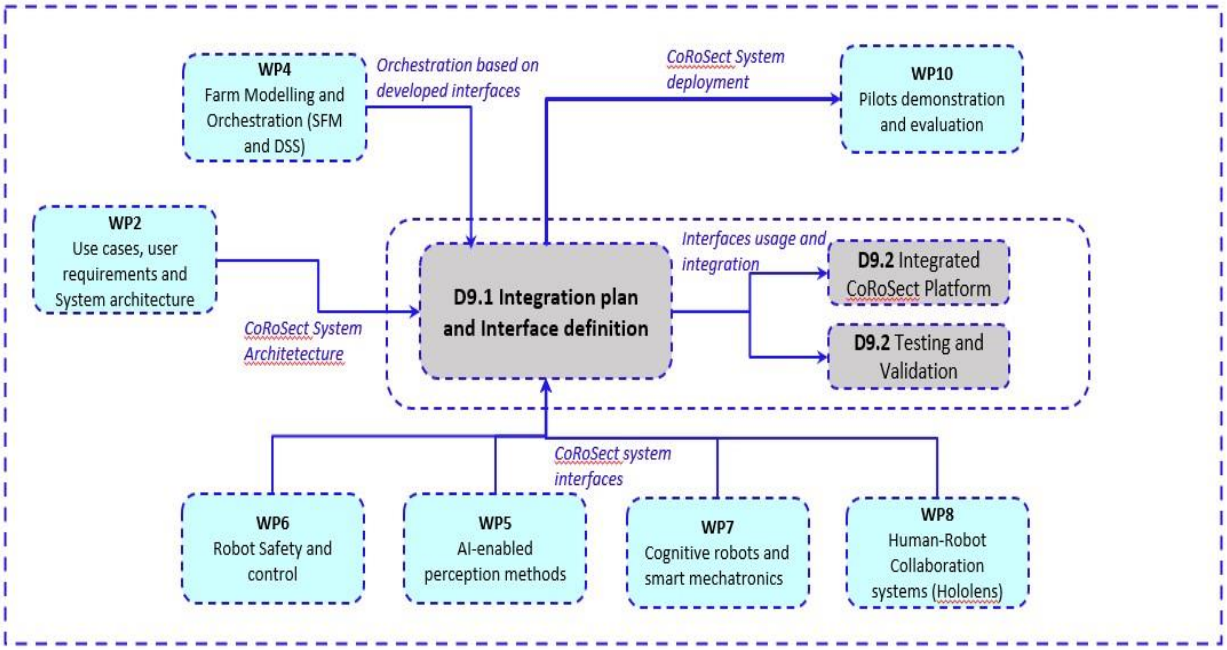


Figure 1 Deliverable 9.1 Relation with other Work packages and Deliverables – Source: HSEL

1.3. Structure of the deliverable

The deliverable requires the specification of the interfaces of the Cyber physical systems (CPS) OT assets with the IT assets, IMS, MES, SFM, DSS. These specifications must follow a set of standards, that the **Industry 4.0** consortium specified in the RAMI 4.0 reference architecture, and the guidelines of the Industry 4.0 platform for developing AAS's and validating the results from these implementations. Hence, the structure of the deliverable is as follows:

Section 2: Integration and Interfacing requirements

- Definition of services and an I4.0 Service Gateway; In order to create compatible interfaces within a service-oriented architecture compliant with the digitalized infrastructure.
- Introduction of the RAMI 4.0 reference architecture; how the insect farming industry can be benefited by following and incorporating Industry 4.0 technologies, for example: how IoT devices such as sensors embedded in I-Crates integrate with other farm's shop floor components to form a coherent system of Cyber-Physical Systems in a digitalized production environment.
- Information about (i) the implementation of standards proposed by the platform Industry 4.0 for harmonizing the interfaces, (ii) the implementation of safety features for human robot collaboration, (iii) the development of AAS's for each asset with standards, sub-models and corresponding files to implement the interfaces before finally concluding on the structure of the interfaces (asset data structure in payloads).

Section 3: Integration and Interfacing approach

- Definition of the integration approach, based on the proposed CoRoSect System architecture (See deliverable 2.4). The approach covers how the interfaces are implemented for each asset, which are the main communication protocols that are used for implementing the interfaces, how is the interfacing with IMS and MES using NGSI v2 API to query, published and subscribed to the context information from and to the assets.

- Brief description of the interfaces developed by the Shop floor component (asset) providers. Based on these, a description is defined then regarding the I4.0 service gateway each component (asset).

Section 4: Conclusion

- This section details the conclusion of the planned integration. It also briefs how the project leverages the benefits of implementing I4.0 standards and also describes the benefits of using this implementation in a relevant use case scenario.

2. Integration and Interfacing requirements

2.1. Services definition

The OASIS RM SOA [4] states that “a service is a mechanism to enable access to one or more capabilities where the access is provided by a prescribed interface and is exercised consistent with constraints and policies as specified by the service description”

Additionally, services in the SoA are defined by their:

1. **Service interface**
2. **Service behaviour**

A Service interface is specified by the syntax and semantics of the call events, and their effects on the information and physical world of the service participants. (E.g., by using Asset Administration Shell).

Service behaviour is specified by one or more interactions (exchange of messages) that happen between components (E.g., Publish/Subscribe (supported by MQTT) or Request/Reply (supported by OPC-UA & REST API)).

Moreover, when interacting through a “service”, the interacting components are called service participants.

There are two roles that service participants can play when being involved in services:

- **A service consumer (SC) role:** when initiating a service call by issuing an operation command to the service provider or by subscribing/requesting the data from the service provider.
- **A service provider (SP) role:** when replying/publishing the data for service consumers to consume or receiving an operational command from service consumers and reacting to it.

NOTE-Participants can act in both roles in parallel with respect to different services.

From the implementation point of view, the role of the service provider is implemented by a:

- **Server/Publisher** –E.g., MQTT Publisher, OPC-UA Server, REST API Server

And the role of a service consumer is implemented by a:

- **Client / Subscriber** –E.g., MQTT Subscriber, OPC-UA Client, REST API Client

A diagrammatic representation of service behaviors and service participant roles can be seen in Figure 2.

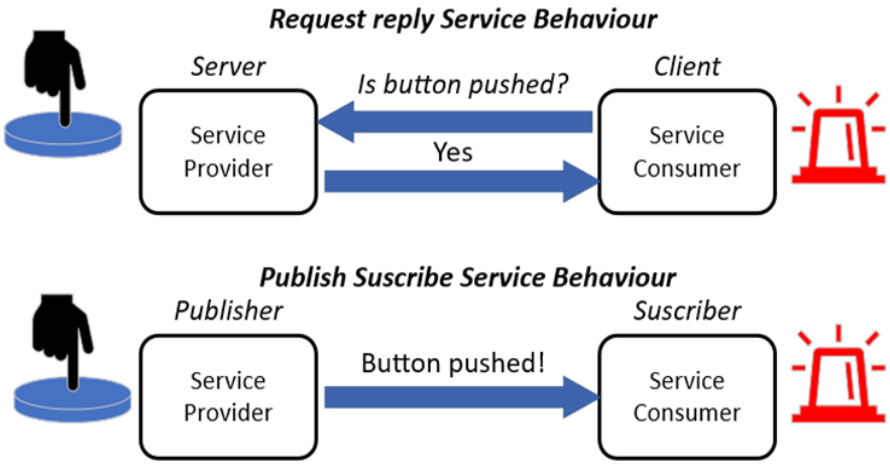


Figure 2 Service Roles and Service Behaviours – Source: HSEL based on [5]

Consequently, based on the description of services and their implementation techniques given above, it should be understood that for each physical asset that would like to operate in SoA manner, a generic I4.0 Service Gateway (Figure 3) which consists of a server/publisher (performing a role of Service Provider) and a client/subscriber (performing a role of a Service Consumer) is required. This service Gateway connects to the Asset Controller/ Gateway using the propriety protocol (defined by the asset provider) but offers the data and capabilities associated to those assets using a defined service-interface and SoA oriented communication protocols like OPC-UA, MQTT or REST API.

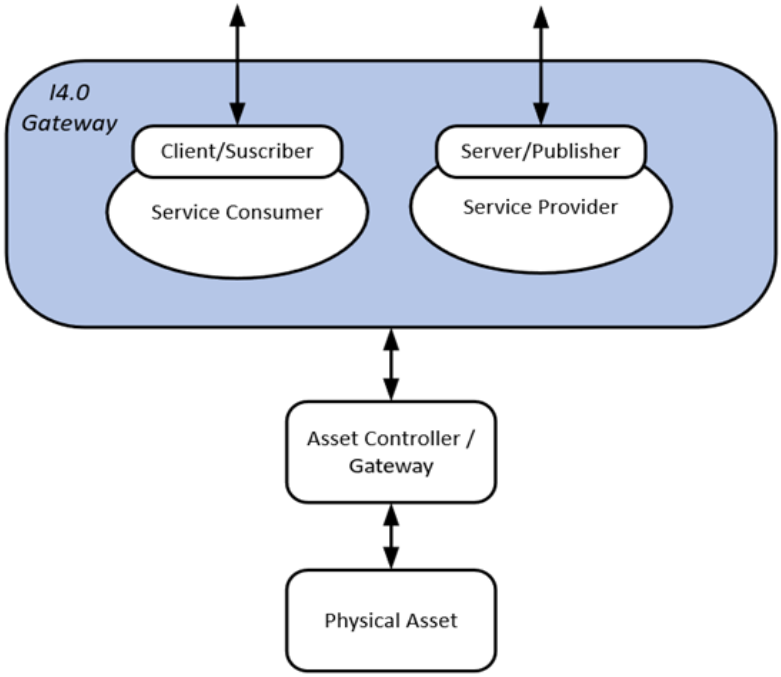


Figure 3 I4.0 Gateway for Each Asset – Source HSEL

2.2. Service Implementation by Asset Providers

From the implementation point of view, since each tech/asset provider (UM & CERTH for M-Robot, CERTH for Augmented Reality glasses (Hololens), ATOS for Object Detector & Route Manager, AGVR for AGV, Robotnik for D-Robot, OAMK for I-CRATES and HSEL for SFM & DSS) will need to implement this I4.0 Service Gateway. Choosing a communication technology through which its Server/ Publisher

will be implemented is needed. These can range from RAMI4.0 Layer 3 compatible communication technologies like OPC-UA Server, REST API Server or MQTT Publisher. However, the Server/Publisher not only needs a communication protocol but also needs a service interface that defines the properties and the operations for which the server would be able to publish data or take requests from service consumers. This is described using the RAMI 4.0 Layer 4 compatible technology i.e., the Asset Administration Shell (AAS). An AAS as a service interface is composed of an AAS description, an Asset Description and a Sub-models Description. The Sub-model is the logical placeholder for containing the description of all the properties and functions for which the asset/service provider would be able to share data or get commands to act on. An internal composition of AAS is described in Figure 4.

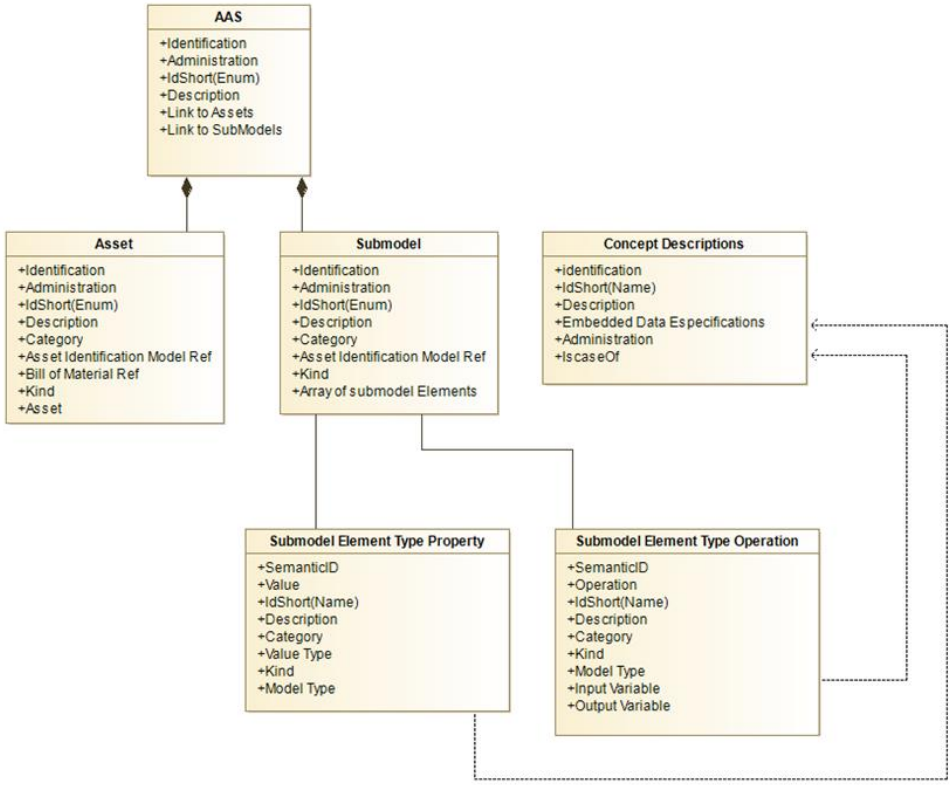


Figure 4 AAS Internal Structure and Relationships – Source HSEL

For the sake of implementation, this AAS is serialized into a compatible format i.e., JSON or OPCUA-Information Model (XML) based on the Service communication protocol (Figure 5). The service interface defined using AAS is also known as a Digital Twin of the Asset because it represents all the capabilities and data offered by the asset (service provider) in a standardized digital format.

The MQTT and REST API uses JSON format to describe their data and functions (service interface) and the OPC-UA uses OPC-UA Information Model (XML) to describe its data and functions (service interface). Thus, the AAS is transformed into respective AAS representation & data exchange formats based on the chosen communication protocol.

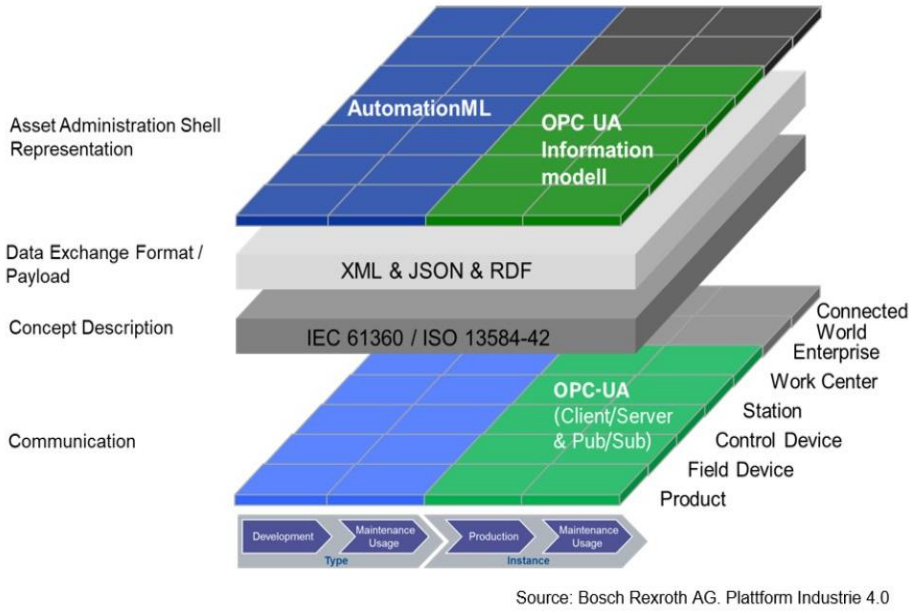


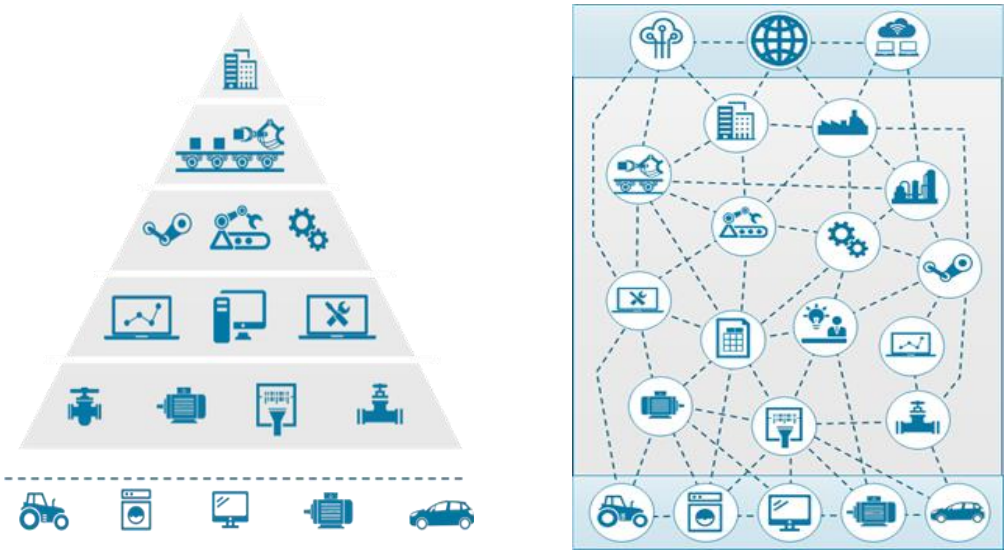
Figure 5 *Graphic View on Exchange Data Formats for the Asset Administration Shell – Source [3]*

2.3. Reference Architecture Model Industry 4.0 (RAMI 4.0)

2.3.1. General Considerations

The aim of this deliverable "implementing a coherent system" relies on assets being able to understand each other through a common language, usually called "I4.0 language", a common communication structure, the semantics of data and its standardization.

Traditionally, the systems that are part of organizations and/or companies maintain a hierarchical structure according to functional attributes. The criteria and considerations of these structures could be defined using ISA 95 / IEC 62264 [6] and ISA 88 / IEC 61512 [7] as a reference. For proper operation, a hierarchical structure is required (Figure 6 a), but at the level of information flow, Industry 4.0 proposes a more harmonious integration between the different elements that are part of that company/organization. It seeks the interaction of systems in which there is no hierarchy for the exchange of information (Figure 6 b). This implies, for example, that a production system can exchange information with a plant team to fulfill a given requirement (based on needs) for a well-defined business.



(a) (b)

Figure 6 Traditional structure (Industry 3.0) and decentralized structure (Industry 4.0) - Source [2]

Some limitations are in Hardware and Software components and their functionalities, functionalities that do not allow data and information portability and scalability of the limited services that these components offer. The process flows follow a set of predefined sequences, the systems are not reconfigurable on the fly and take a lot of time to make adaptations to new processes. The missing functionalities have a direct bearing on the efficiency of production in today's insect farms. Thus, to have these missing functionalities covered, digitalization and networking of components in the OT and IT level is performed. There is also the business part where the functionalities of components are used for business.

Any digitization process, under the principles proposed by Industry 4.0, requires a reference framework. These frameworks usually lay the foundations on which to build the different implementation structures on which a specific business is developed through prerequisites defined by different use cases. From the digitization point of view, and in order to develop an appropriate service orchestration scenario and somehow standardize the digitization process, it is proposed (among other existing frameworks) the use of RAMI 4.0 (Reference Architecture Model Industry 4.0) that is shown in the Figure 7.

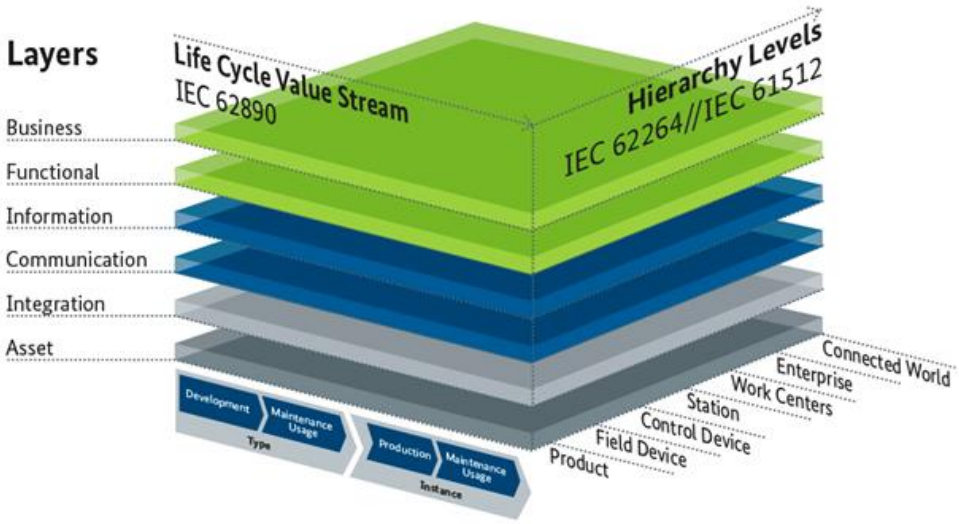


Figure 7 RAMI 4.0 (Reference Architecture Model Industry 4.0). Source [2]

As can be seen in the Figure 7, RAMI 4.0 is a three-dimensional framework composed of three fundamental axes:

- ***Lifecycle Value Stream Axis:*** Based on IEC 62890 [1]. Which considers the aspects referred to the life cycle of the different Assets considered in the digitization process.
- ***Hierarchy levels Axis:*** Based on IEC 62264 [8] and IEC 61512 [9], both also related to ISA 95 [6] and ISA 88 [7] respectively. This means that the structural aspects in terms of functional structures, raised in Industry 3.0, are considered to build this axis.
- ***Vertical Axis (Layers):*** This axis proposes how to approach the digitization process once the potential asset(s) have been identified to cover the specific requirements defined by the different businesses or needs defined in advance. As can be seen, this axis includes all aspects related to the integration of the physical world with the digital world (integration), the form of communication of the information (Communication), the structure of the information (Information), the required functions (Functional) and the business(es) defined within this implementation structure (Business).

To connect a physical or digital thing in the digital world, the RAMI 4.0 provides the concept of the Asset Administration Shell (AAS) in order to create an I4.0 Component. This element is a composition of an Asset and its AAS and is the basic element of a Service-Oriented Architecture compliant digitalization infrastructure. An asset can be a sensor, a robot, a production cell, or the whole farm itself, which can have its own AAS. This AAS serves as an interoperable digital twin that can share the static and dynamic information associated with the asset, with any other networked AAS, using a I4.0 standard protocol, a standardized structured form and defined formats (E.g., MQTT, OPCUA, REST).

The introduction to RAMI 4.0 has already been made in Deliverable 2.3, section 2.1.2, so it is suggested to refer to that document for more details.

2.3.1.1. Lifecycle Value Stream Axis (IEC 62890)

From the digitization point of view, it is essential to consider all the life-phases of the assets during their life, from the time they are created from a purchase order, by a partner in the value chain, to their final disposal by the same or another partner in the chain. The Figure 8 shows a typical life cycle as provided by the IEC 62890 [2].

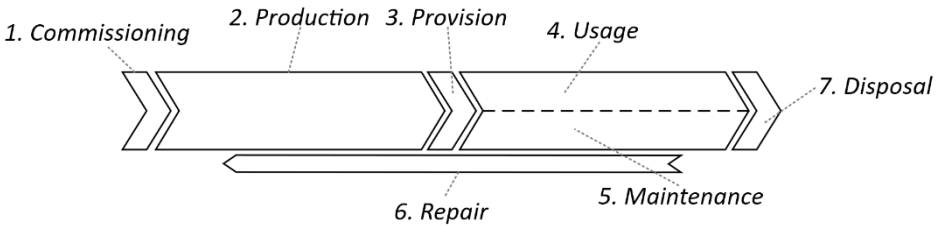


Figure 8 Life ("vita") of an Asse - Source [2]

As can be seen in Figure 8, the life cycle of an Asset can be generalized into 7 stages (From Commissioning to Disposal). Each of these stages corresponds to a specific moment in the life of an asset within an asset's life cycle.

Considerations about the Life-Cycle of an Asset, are very important to the decision-making process, because in each stage of the Asset, the information can be interconnected and can be related to obtain a certain purpose and improve the process.

From the interaction of the information of the different stages of the life cycle of an asset, a wide range of possibilities arise to establish new business between all the partners of the value chain, exchanging in this case, valuable information of the asset within a service-oriented architecture.

An asset, in the case of this project, can be considered as any object that has value for the company, from an insect, the equipment that is part of the farm, the robots, the operators, the sector in general, the whole factory, etc.

The IEC 62890 [2], proposes interesting strategies to address the different life cycles of products. The standard also considers the different life cycles of the components that are part of the products. Based on this, information of components that are part of a robot, for example, can be considered, individualized and exposed in a service-oriented architecture, each with its specific life cycle and related partner.

2.3.1.2. Hierarchy Levels (IEC 62264/IEC 61512)

This axis of RAMI 4.0 considers the organizational structure based on the DIN EN/IEC 62264 [8] and DIN EN/IEC 61512 [9] standards, both of which are derived from ISA 95 [6] and ISA 88 [7] respectively.

RAMI 4.0 proposes this axis in order to correctly position the different assets within the organization of which they are a part. Although the Industry 4.0 approach proposes flexible structures, the hierarchy is necessary to know how the interaction of information and the different criteria for grouping assets must be established for the different use cases.

2.3.1.3. Vertical Layers

The approach to the vertical layers of RAMI 4.0 will depend on the positioning of the assets in relation to the horizontal axes referring to the life cycle and the hierarchy axis.

The procedure to address these layers is a complex process that depends on many criteria and considerations of the defined use case. Each of the layers (Business, Functional, Information, Communication, Integration and Asset. See Figure 7) defines a standardized way and covers a specific functionality to address each of the aspects that are necessary to cover the digitization process of a given asset.

The way to go through the layers can be *bottom-up* or *top-down*. Generally, the most recommended way is *top-down* as it starts from the needs of a particular business and discovers the assets needed to satisfy them.

The way in which the layers are to be addressed can be found in the DIN SPEC 91345 [2].

2.3.2. Asset Administration Shells

2.3.2.1. Definition

Based on [3], the Asset Administration Shell (AAS) is “standardized digital representation of an asset” and has the following attributes:

- **Integrates the asset into the Industry 4.0 communication**, providing a standardized and secure communication interface.
- **Is addressable in the network** and uniquely identifies the asset (e.g., through a unique Identifier like E-Class or similar).
- **Allows controllable access to all information** of the object that represents.
- Can **integrate different kinds of assets** structuring the information in a common way.
- **Maps the entire life cycle of products**, devices, machines and plants through different representation of the assets that the AAS represent. (It means through different Digital Twins conforming a Digital Thread of a specific digitalized Asset).
- It is composed of a well-defined and **standardized structure** in order to structure the information.
- The Asset Administration Shell is the proposed way to **create I4.0 components**.

2.3.2.2. Mapping the AAS in the Reference Architecture Model Industry 4.0 (RAMI 4.0)

According to the vertical layers of RAMI 4.0, the Asset Administration Shell (AAS), can be defined in the "Information" vertical layer, exposing ("Communication" Layer) in a structured, common and specific way, the functions ("Functional" Layer) of the given use case ("Business" Layer). Then, for this information to be enriched and integrated into the implementation structure proposed from the asset(s) in question, the "Asset", "Integration" and "Communication" layers must be defined. As mentioned above, the AAS covers a certain layer of the RAMI 4.0 (Information), but since the information is somehow interacting in all layers, it is possible to perform an analysis of how the AAS interacts with the layers proposed by the reference model adopted for the project (See Figure 7).

- *Layer 1 – Asset:* Covers the representation of the asset that is placed in the hierarchy level axis and the life-Cycle axis.
- *Layer 2 – Integration:* Corresponds to the integration of physical asset into the digital world (AAS as a digital twin).
- *Layer 3 – Communication:* How can the data be accessed in the internet world, in industries it is the mechanism, the protocol used for transferring data between components.
- *Layer 4 – Information:* is the information layer where the standardised data models, information packages are processed and make it available as useful and valuable information in the form of services.
- *Layer 5 – Functional:* Defines the functionalities that are required to fulfil the defined business use case through the defined assets.
- *Layer 6 – Business:* In this case, the digitalized and networked asset is able to perform service-oriented business applications using the functions provided by the Layer 5 and the digitalized data and information contained in the Layer 4. Moreover, as applied in CoRoSect project, the AAS must be exposed in a common and standardized way so that orchestration is possible and an intelligent decision-making process for insect production can be defined and executed.

In the implementation addressed in CoRoSect, the light weight Information Management system collects the context data from the interfaces by using a set of API’s, the information is then served to both the OT and IT levels, including also the connected world (supply chain and collaborative enterprise networks).

2.3.2.3. Logical View of the AAS

As described earlier, the digitalised assets will communicate their services (data and information including functions/applications) in a common I4.0 network. This will flatten the hierarchy as the assets in IT and OT layer will now be able to communicate in a coherent network. The outline of the I4.0 network is shown in the Figure 9.

Figure 9 CoRoSect I4.0 Network view (AAS perspective) - Source Deliverable 2.3

2.3.2.4. Specifications of AAS

The information stored in the AAS should be exchanged in a meaningful way between assets and or components, so this information has to be first processed and structured. In order to make specifications there are a few structural principles that need to be detailed:

- Information metamodel of AAS and its Sub-models.
- Exchange formats for information transport.
- Identifiers.
- Access control.
- Mapping of AAS to suitable technologies such as OPC UA, MQTT, JSON, XML etc.

Information metamodel of AAS

Metainformation Model of the AAS describes how the information needs to be modelled, it is as stated in Figure 10:

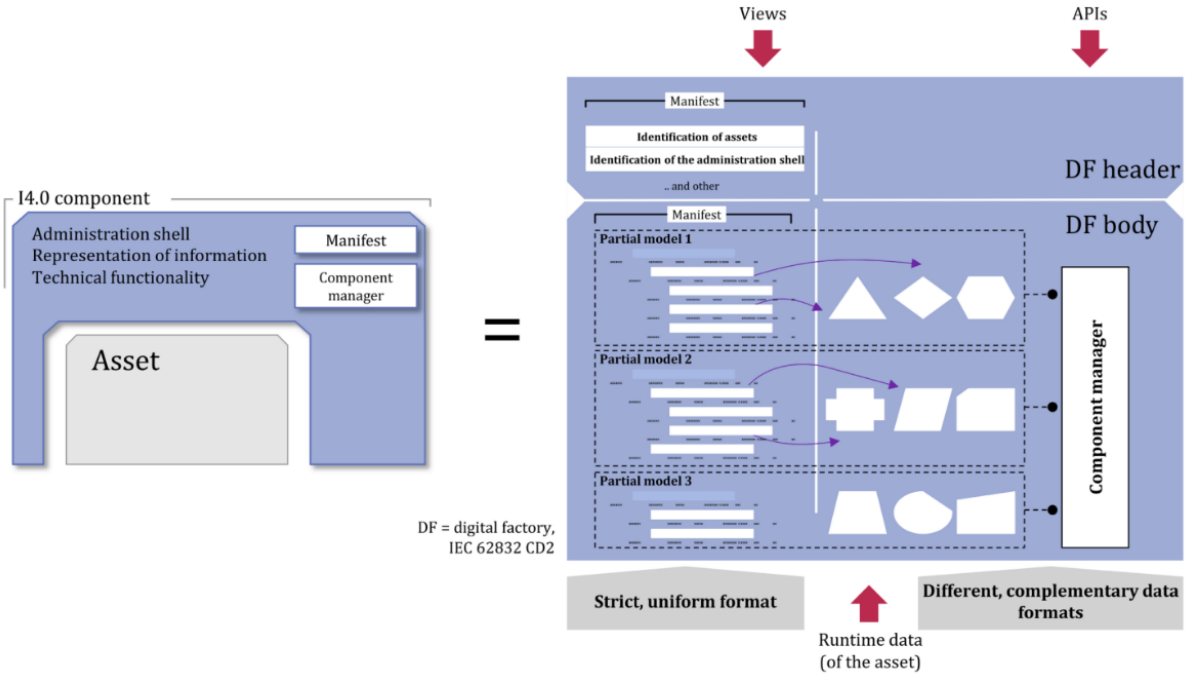


Figure 10 Metainformation model of AAS - Source: [2]

The AAS information model is therefore independent of technology, it's just a representation of data in a standardised structure. Information model just specifies how to model the information.

Exchange formats

The AAS information exchange between components via file exchange, can be done in 3 ways as depicted in Figure 11:

1. File Exchange – A standardized file like JSON or XML, inside is some valuable information stored, this is passive AAS
2. The information model is also the basis for exchanging information via a standardised API. In this case it is reactive AAS, where a set of API's call for information to be exchanged.
3. A more pro-active approach is the exchange of AAS via a secure I4.0 communication network and protocol as an AAS itself.

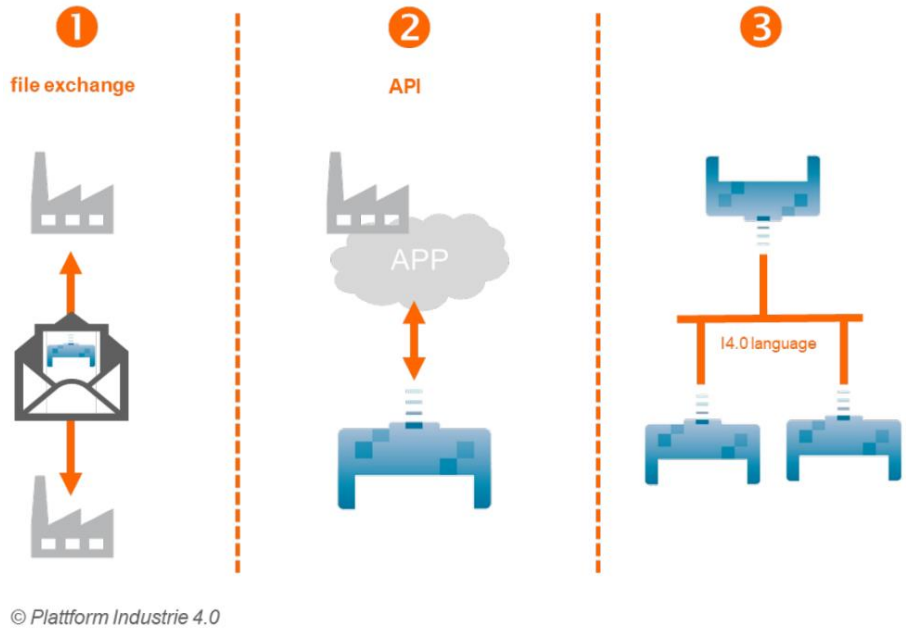


Figure 11 Types of Information exchange via Asset Administration Shells – Source: [3]

In this case, the asset administration shell data is accessed using a specific API that will be part of the implementation.

Identifiers

Identifiers are required for unique identification of elements within the smart manufacturing, the insect farm Assets digital representations are to be identified uniquely, especially identification is required for:

- AAS's
- Assets
- Sub-models
- Sub-model elements
    - Properties
    - Operations
    - Etc.

This unique identification helps in accurately placing the data to relate for the interfaces to identify and place the context data. The linking of data for various other entities to identify is very critical for exchanging information via the 3 methods detailed above.

Access Control

The integrity of the AAS should be considered depending on the requirement. The properties, data and functions will contain information that all value creation partners or components might not require. Hence, the structure of AAS should take into account the aspects of security, access protection, visibility, identity and authorization. A "no security" status can also be implemented if the risk analysis permits it.

Mapping

For the realisation of a "Novel digitalised integrated robotic solution based on Reference Architecture (RAMI 4.0)" sharing of information between different systems is crucial. For usage of data in different life cycle phases of an asset different data formats are recommended to be used [3]. For each of these format's serializations and mappings of AAS are provided as a JSON and XML format. The shop floor component providers will use this serialisation file of each Sub-Model of their asset to publish information via the OPC UA or MQTT server that are built on their machines.

Each shop floor component has their own servers that connect to the Data Management system (IMS) via a secure end point. In general, for applications to be connected together a connector is needed. The IMS connects the OT and IT levels of the farm using a set of standard APIs. The API uses the JSON Payload published by the asset servers. The received context data is then modelled, processed, stored in the IMS system. The context broker then Publishes/subscribes to data on demand to and from the Shop floor manager guided by the Decision Support system. The IMS therefore manages the Digital twins (Through each AAS) as Digitalized Information Models.

The IMS and its implementation can be reached in Task 4.3 and its corresponding deliverable. A brief overview will be given in Chapter 3 of this referred document.

2.3.2.5. Required elements of the AAS

Sub-Models

AAS is made up of a series of Sub-Models. These represent different aspects of the asset concerned, the aim is to standardise one Sub-model for each aspect [10], for example: the aspect of "safety" has one Sub-Model, so does condition monitoring, nameplate, technical data, communication etc. There are many standards that can be used to describe the attributes, properties and information that will need to be stored to describe the different Sub-Models of an Asset. It is recommended, whenever possible, to base the information defined according to the standards in order to ensure interoperability and language compatibility between the different partners in the chain.

Each Sub-Model of the asset has its own ID defined as IDShort, the standardised Sub-Models contain a structure quantity of properties that can refer to data and functions, these properties are the constantly readable directories of key information. The standard Sub-Models implemented for the shop floor components are as follows:

- ***Nameplate**. Used to describe general attributes and properties of a certain Asset like, manufacturer related data as an example.*
- ***TechnicalData**. Contains technical data related to the asset in question.*
- ***OperationalData**. Sub-Model on which different parameters can be defined on the assets. Also defined to store specific data related to real-time measurements.*

- ***OperationCapability***. *Used to store the Asset's available operations. The properties defined in this model, called operations, are created to define the different inputs and outputs.*
- ***AssetConditionMonitoring***. *Sub-Model used to perform condition monitoring.*
- ***BillOfMaterials***. *Aims at sharing information in an interoperable way providing a view of contained material that composes the asset.*

The Sub-Model and its elements such as properties, files, operations, etc. must be clearly identified with a unique identifier. Global identifiers are defined in the ISO 29002-5 [11] (e.g., eCl@ss and IEC 61360 Common data dictionaries) and URI's (e.g., Ontologies)

Properties and Operations

Properties and operations (property that admits inputs and outputs) are objects that generally form part of the Sub-Models and they are the ones that usually store punctual values or functions as end-points (operations). Properties are the set of data required to be structured and serve as a function of the use case(s).

Concept Descriptions

Based on the "Details of the Asset Administration Shell – Part 1" [3], a Concept description can be defined as an "unit of knowledge created by a unique combination of characteristics".

This implies that these attributes are of particular importance when a specific knowledge value is to be assigned to a property or element (Sub-Model). For example, a property called "Temperature" can be defined and its corresponding concept description would be the unit of measurement (e.g., °C or °F).

2.3.2.6. RAMI4.0 compliant Interfaces

Implementation of standards for communication of asset data need to comply with Industry4.0 standard protocols, such as OPC UA, MQTT and Web services (through REST API, for example). The options are limited to communication types that allow information sharing within a service-oriented architecture. Within the context of Industry 4.0 and a service-oriented architecture, these protocols and forms of communication are expected to have specific attributes, such as flexibility, atomization, abstraction, autonomy, discoverability, standardized service contract, reusability or composability, among others.

Figure 5 shows the different exchange formats supported by RAMI 4.0. JSON is commonly used in MQTT and Web Services (REST-API) and in OPC UA, XML (OPC UA Node set) is often used. In any case, the final exchange formats will depend very much on the implementations of each of the data exchange protocols/formats.

In this project, the OT components, the components at field level such as D-Robot [#9 Robotnik], M-Robot/VI [#1 UM], Augmented Reality glasses (Hololens) [#2 - CERTH], will use OPC UA as a protocol for implementing the interfaces, the rest of the assets such as AGV [#10 AGVR], I-Crates [#5 OAMK], Route Manager and Object detector [#8 - ATOS] will use MQTT as a communication interface. The IT components like IMS, DSS, SFM will use Webservices (API's). All these assets will communicate to each other by using IMS component, the context broker.

2.3.3. Industry 4.0 Standard Implementation

One of the core principles/visions of this implementation of I4.0 in CoRoSect is "to give its components and production processes the capability to adapt to production needs and more efficient allocation of resources". This is based on a number of emerging technologies such as IoT, Artificial intelligence, Robotisation and the concept of Service-oriented Architecture. To say, a production

process is industry 4.0 compliant a set of attributes/functionalities needs to be associated with the processes such as, portability (data), system interoperability, product customisation, collaborative robots and robot cells, interconnection and security, Artificial intelligence adoption. How the CoRoSect´s structure effectively applies these concepts will be discussed in detail.

2.3.3.1. Portability and Interoperability

With the creation of the Assets administration shell for each Asset that is part of the project, the creation of an atomized I4.0 component is achieved, which allows to expose basic functionalities in order to make the orchestration process possible.

Portability is achieved by using standardized information exchange formats such as JSON and XML, which can be serialized and deserialized, for the interaction of the information they expose. In turn, the services for information access are performed in particular platforms in the cloud (FIWARE [12], BaSyX [13]), but the information exchange is done through standardized protocols which are compatible with the concepts that Industry 4.0, using RAMI 4.0 as a reference, proposes.

Interoperability is achieved by using compatible forms of information exchange between the different Assets. Since there is a great diversity of Assets, such as robots, Augmented Reality glasses (Hololens) [#2 - CERTH], etc., it is necessary to incorporate components (NGSI) in order to adapt, make compatible and centralize the exchange of information between all the common elements in the network. Then, through the different IoT agents, the interaction of information is achieved through the different Asset Administration Shells that represent specific information of each asset.

More details on how interoperability and portability between the different assets, that are part of the project has been achieved, can be found in deliverable "**Deliverable 9.2 - Integration CoRoSect Platform I**".

2.3.3.2. Product Customisation

The concept of flexible production requires a service-oriented architecture. This implies that the assets that are part of the organization should have the capacity and the ability to expose their basic functionalities through a common network (that could be local or in the cloud).

Then, in the cace of possible variations in a given product, the production system can adapt (within a certain range) to possible variations in demand.

This is also linked to reconfigurable systems, i.e., that support parameterization based on different strategies, such as artificial intelligence, neural networks, etc., i.e., parameterization strategies based on data Analytics. Parameterization processes can also be defined in multiple ways. Therefore,t by using data Analytics, multiple advantages could be obtained.

2.3.3.3. Collaborative Robots and Robot Cells

By combining the capabilities and new functionalities obtained with service-oriented architectures and in combination with flexible production systems, the concept of collaborative work can be defined among the assets that are part of the project (Organization).

Particularly robots, whether they collaborate or working in a fixed cell, within a service-oriented structure, can adapt and reconfigure themselves according to the requirements of their "environment". Environment in this case refers to all the services found in the common network accessed by the robots.

Thus, a correctly digitalized Asset, working in a service-oriented architecture, is sensitive to what is happening in its environment, which allows it to execute and consume "Available Functions". Here

the mistake should not be made of defining it as “automatization”. Automatization itself is already performed by the robot(s). With the digitalization process, the necessary support is given to guide in the best way all the resources for the defined business (through the use cases) where these Assets perform their functions.

2.3.3.4. Security considerations

Referring to safety considerations and the IEC 61511 [14] and IEC 62061 [15] standards, the following work packages are mentioned:

- Task 6.7: Safety concept for robotic systems (planning).
- Task 6.8: Safety concept for robotic systems (documentation).
- Task 6.9: Safety concept for robotic systems (final).

In the deliverables corresponding to these activities, bearing in mind that some of them are under development, the specifications required in this aspect can be found.

From the point of view of digitalization, the proposed orchestration of services can be very useful to prevent risk situations by introducing specific functionalities for this purpose. For this purpose, specific functionalities can be defined as attributes or elements of a Sub-Model within an asset administration shell, in order to structure the information required for a possible security orchestration to support the control systems, if any. Otherwise, security issues can be implemented in various ways according to the asset's ability to communicate and present information [2].

2.3.3.5. Considerations based on IEC 62453 and IEC 61804

From the point of view of Industry 4.0, the integration of the different components (Assets) that are part of the CoRoSect project, must be carried out using specifications compatible with the reference architecture on which the digitalization process is based (RAMI 4.0).

From the Industry 4.0 point of view, the integration of the different components (Assets) that are part of the CoRoSect project must be carried out using the specifications compatible with what is proposed by the reference architecture on which the digitization process is based (RAMI 4.0, see Figure 7). By establishing the correct specifications of the integration, communication and information layer, and having previously defined the business and functional layers, any asset, regardless of its organizational hierarchy, can be interconnected with others and expose its functionalities in a service-oriented architecture.

On the other hand, through the definition of the Asset Administration Shell (AAS), it is possible to structure and standardize the information of the representative data of the Asset, through its digital Twin, for the specific use case of this project.

In turn, through the definition of the Asset Administration Shell (AAS), it is possible to structure and standardize the information of the representative data of the Asset, through its digital Twin, for the specific use case of this project. This would cover the so-called "Information" layer provided by RAMI 4.0.

By using MQTT, OPC UA and REST-API technologies, the information is made available in a specific and particular way in a common network (local or not). This would cover the so-called "Communication" layer of RAMI 4.0

Then, the different elements are required to cover the "Integration" layer of RAMI 4.0. Bearing in mind that in this layer the information is transmitted from the physical world to the digital or information world, the additional device(s), if applicable, of each asset must be added and/or configured in the

implementation stage. This should be considered by the different partners that are part of the CoRoSect project, taking into consideration the specific statements of the IEC 62453 [16] and IEC 61804 [17] standards.

2.3.3.6. Analytics to improve the decision-making process (Based on SoA)

Upon successful completion of the digitization process, a structure of information and services is obtained that ends up supporting specific businesses by covering specific needs or requirements according to the use case. But always, every digitization process ends up satisfying a real need, which is actually the reason for carrying out such process.
On the other hand, the service orchestration process and data analytics play a fundamental role in the decision-making process. Service orchestration must be based on an analytics-based logic Data analytics must provide support based on facts (information), which may come from multiple sources, to make decisions in real time. Under this concept, each asset, regardless of its type (Robot, MES, AGV, Insect Box, etc.), can make concrete decisions in order to satisfy certain requirements/demands that may come from different sources (Other assets, human, market, etc.).

For this reason, the service-oriented architectures intended to be used in this project, requires a common language for the exchange of information (generally provided by the different interfaces and gateways) to enable the orchestration process.

The aforementioned is related to the following project tasks/deliverables:

- **Deliverable/Task 4.1 Adaptative Farm Process Modeling**
- **Deliverable/Task 4.2 Data Analytics to obtain prediction models**
- **Reference Deliverable/Task 4.3 Orchestration Engine Implementation**

2.3.4. Guidelines - Structure of data and Compliance (HSEL)

2.3.4.1. Information structuring

For digitization of shop floor assets, AAS has been chosen as the digital twin component which is a standard set by the RAMI 4.0 framework. The AAS addresses the information structure and the formats compatible for integration of assets in the OT as well as IT layers.

For this reason, the previous sections have been dedicated to establishing the conceptual bases necessary for the definition of the structure of the asset information with the AAS. Then in the following sections, the strategy used for the implementation of AAS in a service-oriented structure with multiple interfaces and connections will be discussed in detail.

Within the elements available according to the reference document [3], the elements defined in section 2.3.2.5 are used since they are sufficient to describe the information required for this specific use case.

Then, because the project has different partners, who will be dedicated to the actual implementation of the assets with the required interfaces for their subsequent orchestration, it is decided to generate a spreadsheet from HSEL that facilitates the definition of the parameters that are required and that must be available in the AAS.

Once the file for each of the assets has been generated, the members of the HSEL team will generate the information that the AAS must contain using specific software which will be described later. Depending on the type of implementation from the asset's point of view, the structured information will be exported in well-defined formats (JSON and XML in this case) in accordance with the structure defined in the reference document [3]. It must be highlighted that each partner, that makes use of

these files, respects the structure of the information provided in terms of the nomenclature used according to the standards and definitions adopted from HSEL.

The information flow can be seen in figure 17 that will be displayed in section 3.3.2.

2.3.4.2. Formats used for communication

From the point of view of digitization, under the RAMI 4.0 framework and also based on the information provided in the document [3], there is no specific methodology defined on how the information that is being offered under this framework should be implemented.

As long as the semantics and syntax of the information provided is respected, each user can dispose of the information. From the implementation point of view, this is the methodology that is most convenient for the specific applications that are being carried out. As an example, a complete Sub-Model of an AAS can be published as a topic on an MQTT broker. Thus, each subscriber accesses the complete Sub-Model, having to process it in a specific stage to extract and finally dispose of the information. Another case that can be cited is the use of information nodes of an OPC UA server that is created from the information models supplied (previously generated with the tool that has been used to generate the AAS). For example, OPC UA methods can be used to define the operational states of each of the assets that are part of the implementation.

Therefore, it is recommended to use good practices regarding the implementation of information models, as long as the structure and definition of the information that has been provided by HSEL through the AAS is respected.

3. Integration and Interfacing approach

3.1. Introduction

The basis of implementation of CoRoSect for insect farms follows a certain standardization methodology to integrate farms into a coherent, safe and scalable environments. According to the OASIS RM SOA, the I4.0 gateway is the core of implementing a service-oriented architecture [4]. The service interface and service behavior form the basis for this implementation. Based on the requirements defined in sections 2.1 and 2.2, the desired interfaces needed to manage, orchestrate, and store the data are designed and the implementation of services by the assets providers using the service interface (AAS)is specified. The generic I4.0 service gateway is implemented using MQTT, OPC UA or REST by the asset providers on their premises/cloud for publishing data to service consumers. In addition, for a system to be coherent, there might be data that the asset needs in order to execute certain actions. Therefore, it is required that the I4.0 gateway implements a client- subscriber to act as service consumer in each of the asset gateways.

Though it is relevant and technically possible to have a client/subscriber implemented for each server/publisher (service provider) it needs to have a connection but is cumbersome and limits the expansion of the system. Also maintaining connections to all the service providers (assets) is extraordinarily difficult and not recommended. Therefore, it is pertinent that a central broker is brought into the picture. This central broker (IMS) will be connecting to all the service providers as a service consumer for reading their data and sending a command to execute the functions through the AAS (service interface). This AAS is implemented using OPC-UA Server / MQTT Publisher / REST API Server developed by all tech providers (e.g., UM & CERTH for M-Robot, CERTH for Augmented Reality glasses (Hololens), ATOS for Object Detector & Route Manager, AGVR for AGV, Robotnik for D-Robot, OAMK for IC)

The IMS will also be providing a service (role of service provider) implemented using a REST API Server supporting an NGSI interface. Using Smart Data Models as its modelling language for the tech providers to only implement a single client/subscriber (role of service consumer). That is to send the request for data from any particular service provider (asset) or send a request for initiating/invoking a command on any of the chosen service provider (asset). This is detailed in **D2.4** and **D9.2**.

This effectively means that the IMS is a pivotal point to which any asset, in the role of service consumer, can ask for data or invoke a command on another asset that is connected to it. Likewise, it is a central point to which all the assets are connected in the role of service providers and serve the requests for data or execution of the command. In summary, no two assets maintain a connection to each other but exchange information in various roles of service consumers or service providers through the IMS. (Figure 12)

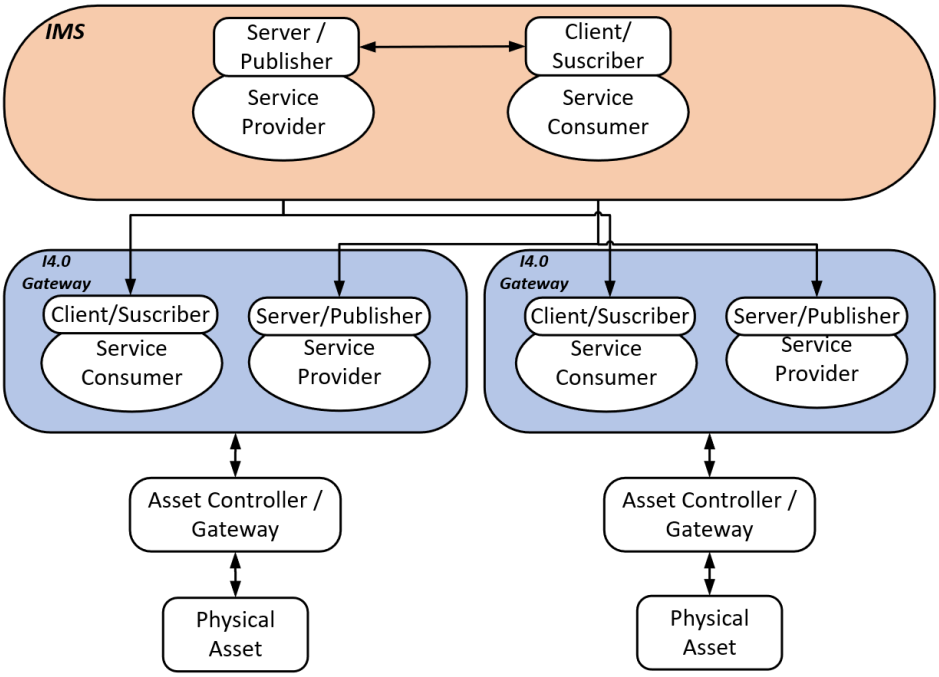


Figure 12 The different assets integrated with the IMS – Source HSEL

The IMS will be a FIWARE [12] Context Broker based integration framework to integrate the assets into a coherent CoRoSect system and execute production recipes in a Service-Oriented Manner.

Thus, with the application of the AAS for defining the service interface and the Context Broker (IMS), the attributes of an SoA based production system can be fulfilled. These attributes are described in Table 1.

Table 1: Attributes of an SoA based Production System – Source HSEL

Attribute	Description	Technology that supports the attainment of the attribute
Flexible	The ability to react to changed requirements or new subsystems	AAS & Context Broker (IMS)
Scalability	No limitations through the size of an automation system	AAS & Context Broker (IMS)
Modularity and standardized interface	Replacing a system should not cause adaption of depending on systems	AAS
Common information model	A similar Information Model describing the information of assets	AAS
Data Visibility	Explicit marking of properties and operations that are offered by the asset	AAS
Historic data access	Provision of accessing the historical data about the asset	Context Broker (IMS)
Inter-enterprise data exchange	Providing facility to extract data within or outside the system	Context Broker (IMS)
Privacy, integrity, and security	Providing facility to extract data with proper privacy and security	Context Broker (IMS
Service detection and orchestration	Seamless integration of participants. If new services are detected, these should be configured and used automatically. Services	AAS & Context Broker (IMS)

	need to be identifiable and orchestrated to support the execution of business logic	
Real-time communication	The need to send and receive data in real-time	***Can be achieved by applying the existing cycle-based field bus communication for a few isolated, critical control functions or services that need to be defined in such a way that they are independent of real-time constraints***

Thus, based on the meaning of service, roles of service participants, service interface (AAS) & service behavior & the Context Broker (IMS), it can be agreed that a common integration approach would look like Figure 13

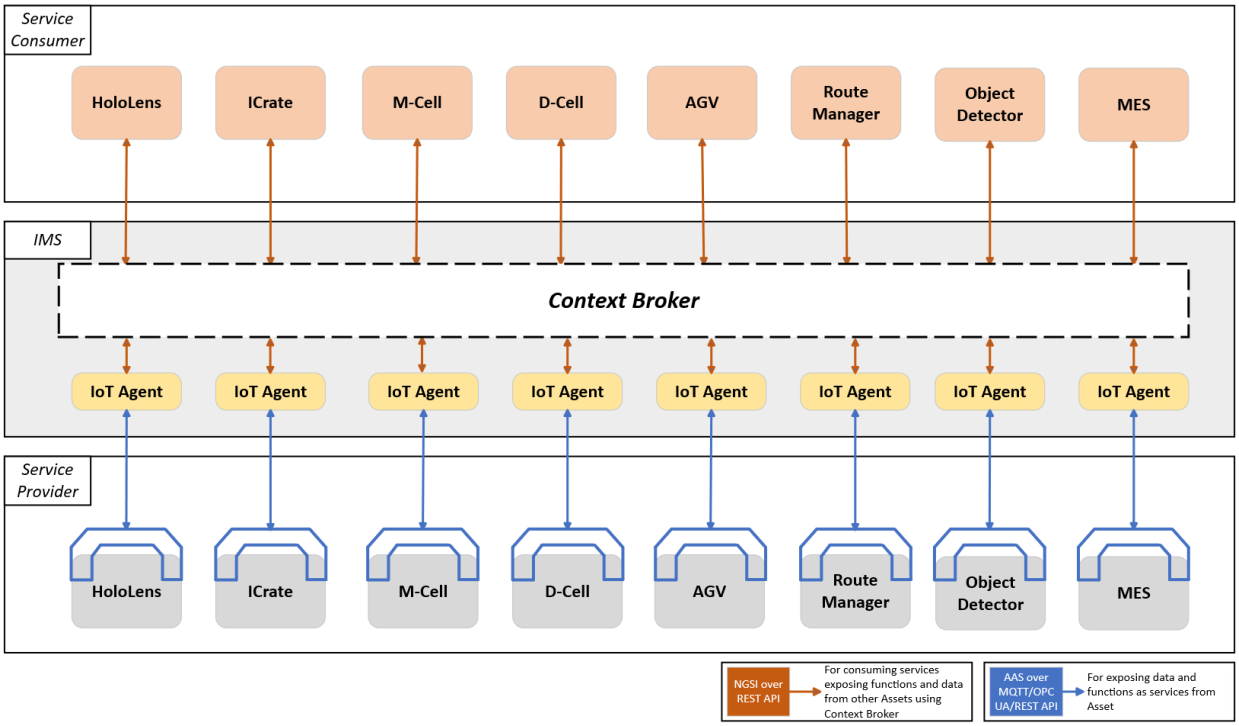


Figure 13 Integration Approach for Assets in CoRoSect – Source HSEL

3.2. Reference Implementation of I4.0 Service Gateway

From the asset provider the implementation of the Asset I4.0 Gateway for each asset that the CoRoSect system consists of would result in Asset gateway containing a server/publisher and a client/subscriber, as depicted in Figure 15. The service interface is basically a JSON file serialized from the AAS package explorer software that will be used by the asset providers to publish the data as per the schema using MQTT/REST API/OPC UA.

The reference implementation of asset I4.0 Gateway is shown in Figure 14

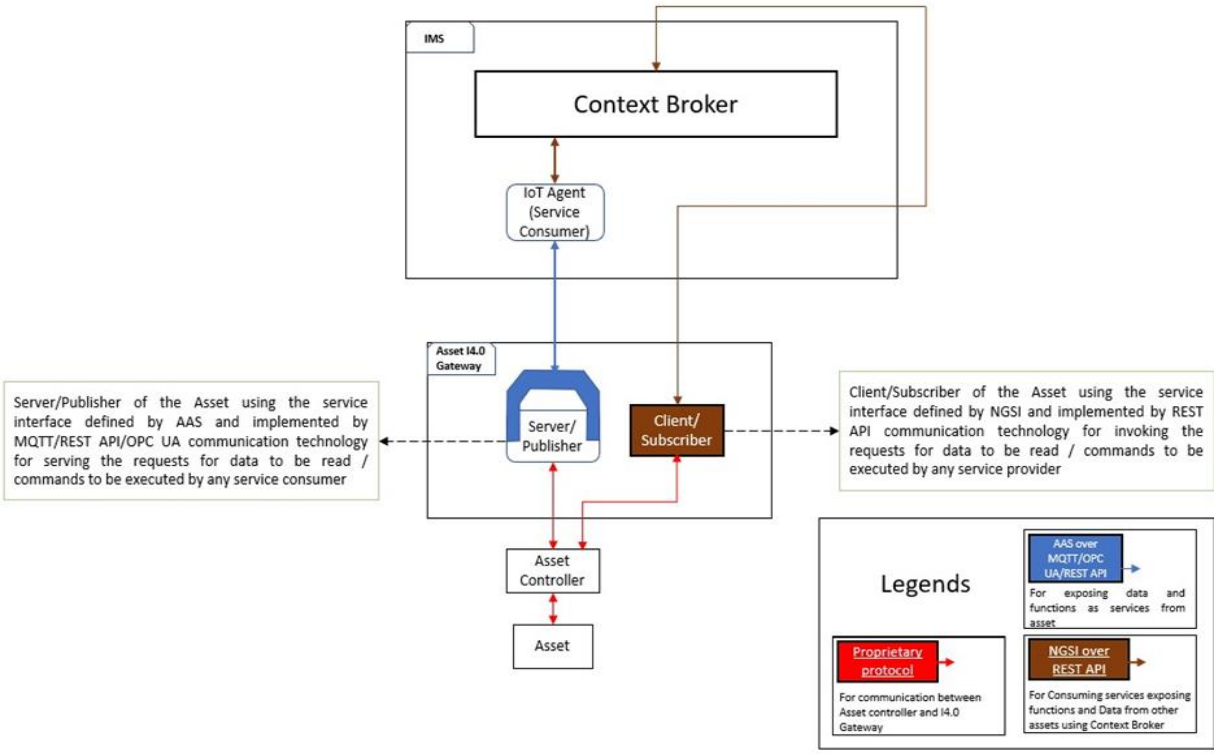


Figure 14 A Reference Implementation of Asset I4.0 Service Gateway – Source HSEL

Section 4.1 describes how to implement a server/ publisher of the I4.0 Gateway described above. The reference implementations are based on OPC-UA, MQTT & REST API Server. The asset providers will have the freedom to choose the protocol based on what can best support the business cases. Section 4.2 describes the implementation of a client/subscriber based on REST API Client. But it is a must that all the assets implement both a) one server/publisher defining it service interface using AAS and b) one client/subscriber for calling services defined in NGSI. This will help in making sure that I4.0 Service Gateway is implemented for all the assets and all the assets are integrated into a single coherent CoRoSect system in a RAMI 4.0 compliant SOA manner.

3.2.1. Implementation of servers/publisher's component of the I4.0 Service Gateway

The OPC-UA Server is the basis of OPC-UA communication. It is a server software that implements the OPC-UA communication stack and thus provides the standardized OPC-UA interfaces to the outside world. There are standard services that OPC-UA Servers support e.g., Read, Query, Invoke, Write etc. The way to implement an OPC-UA Server is described in Figure 15.

An OPC-UA Server Application is implemented using programming languages like c#, java, python, & C. these server applications will have some internal functions and variables defined. These variables and functions on one hand will have a connection to the asset controller using a property connection like ROS, SQL, File Messaging System, Modbus, Profinet, etc. On the other hand, these variables and functions will have a mapping to the properties and functions defined in OPC-UA Address Space. The OPC-UA address spaces is a collection of nodes populated from the OPC-UA Information model file defined using the specifications of the AAS. So, in summary, AAS Service Interface) converts to-> OPC-UA Information model coverts to-> OPC-UA Address Space.

The OPC-UA Server API which is used in the OPC-UA Server Application is provided by the open-source libraries [18] [19] & [20]. The API's implement an OPC-A Communication Stack which is responsible for managing for creating an endpoint for exposing the address space (service interface) to the IMS or external OPC UA test client's like UaExpert [21]. The communication stack will take care of handling request messages that are coming from the IMS and giving them particular response ie. The value of the properties or output data from the execution of functions/commands.

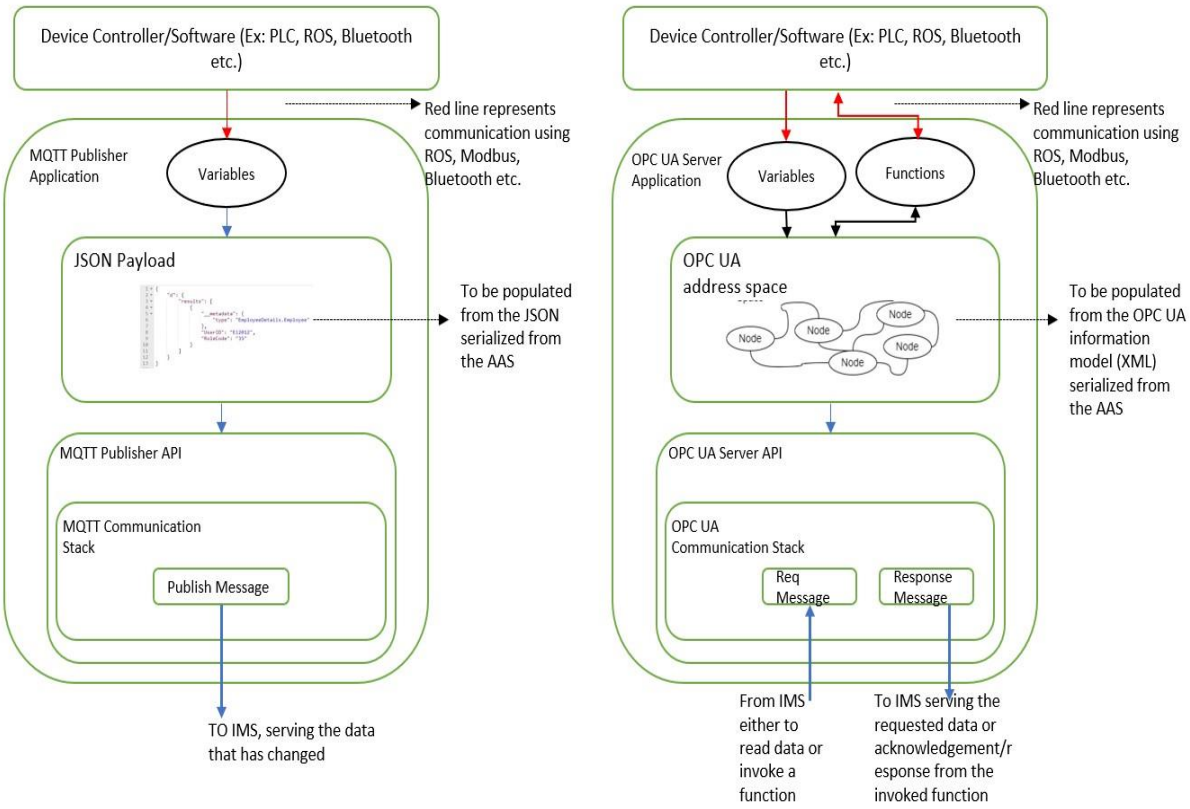


Figure 15 MQTT and OPCUA server implementation – Source HSEL

MQTT Publisher on the other hand is the basis of MQTT communication. It is software that implements an MQTT communication stack that is used to Publish the data to a broker. It supports only one service of publishing the data to the broker. The way to implement an MQTT Publisher is given in Figure 15.

3.2.2. Implementation of the client/subscriber component of the I4.0 Service Gateway

The client/subscriber component of the I4.0 Service Gateway needs to be implemented using the REST API Client. A REST API client will work on the request/reply pattern and will be used to implement the service consumer role for the asset providers. With this client, the asset providers can request any data or issue a request for invoking a command on other service providers (asset providers) using the IMS. The necessary support for implementing the client will be provided by ATOS. But the client has to be developed by the asset provider ex. UM, CERTH, Robotnik, AGVR etc.

A reference implementation of this REST API Client is shown in Figure 16.

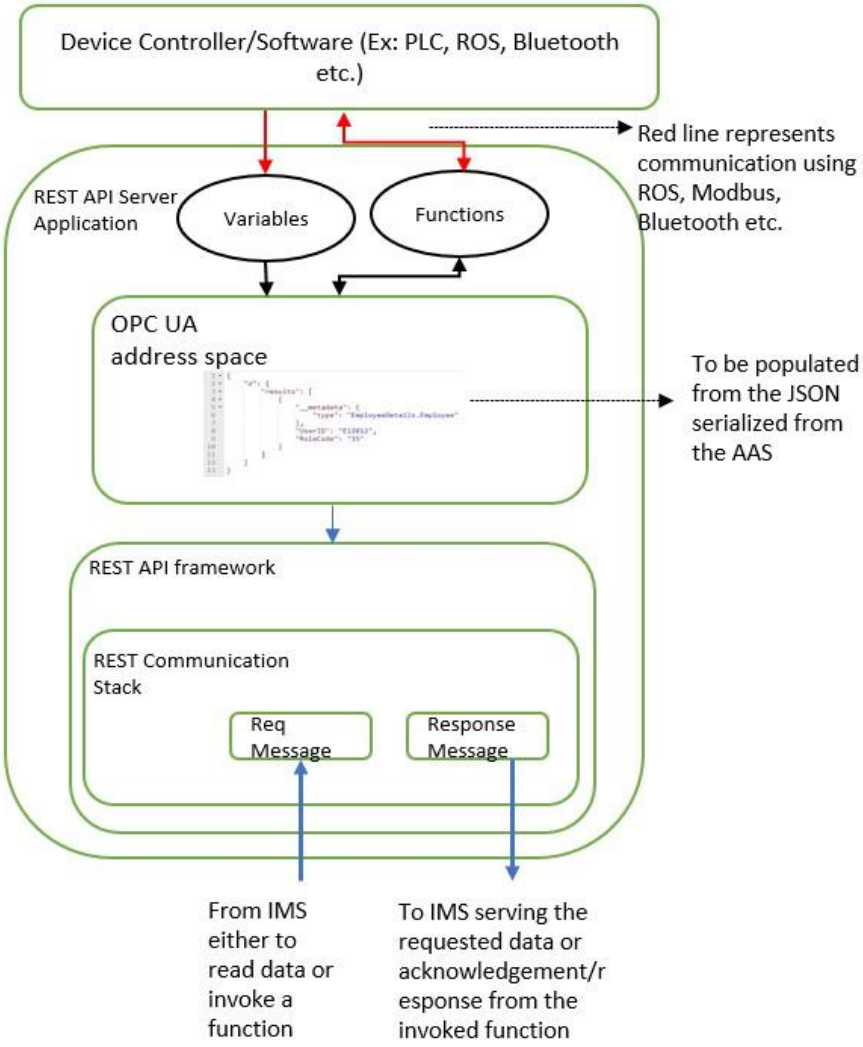


Figure 16 REST API Server Application – Source HSEL

3.3. Interfacing Approach

3.3.1. Generic Schema for assets interfacing

The assets in the CoRoSect system are a combination of hardware and software. For each type of asset the asset administration shell service interface is formulated by a set of Sub-Models and attributes. These data and information in the Sub-Models and attributes are specific for certain assets and depend also on the life cycle stage of the asset. In this implementation the assets are in their Instance phase of the life cycle refers to the usage/maintenance stage. Hence, referring to the literature of the "Details of AAS [3]" gives a clear perspective of what the Sub-Models should be and what attributes must the Sub-Model hold. Each asset provider defines what information should the Sub-Model publishes in its serialisation form and when the publishing should happen. In this project the data that is necessary is published to the IMS via the service gateway, it's also important to know that not all assets of all shop floors are digitalised. As per the common understanding of the Consortium and especially the asset providers, key functional and operational attributes are selected after careful scrutinization and references from standard AAS implementation. The generic scheme of

contents of AAS are mapped in excel form (see Figure 17) and shown in the example illustrated in Figure 22.

As mentioned, each asset has its own set of Sub-Models and corresponding attributes, the AAS is defined by

1. Understanding the Asset
2. Knowledge of interaction of assets with other assets in an insect farm.
3. Data and functions required based on common understanding of interaction of assets.
4. Match the functions and properties with IEC CDD and fetch the semantic data
5. Fill the details from 3-4 in the AASX tool
6. Validating the final schema
7. Transform the data into required format (JSON, XML)
8. Validate the final format.

3.3.2. Information flow for interface generation

Figure 17 depicts the AASX information/definition flow. The AASX hence is the digital representation of the functions and operations the asset is capable of producing and consuming.

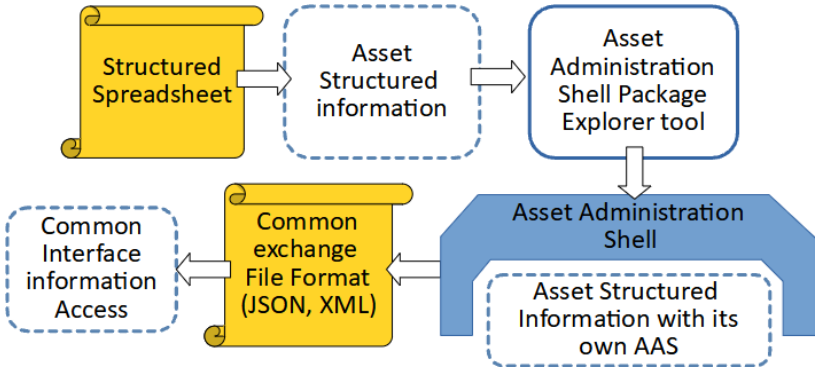


Figure 17 Information flow for Interface creation – Source HSEL.

3.3.2.1. AAS deployment method

This section illustrates the AAS deployment processes in the edge layer, i.e., how to integrate the AASX model into the OPC UA information model and MQTT and how to map the AAS model data with the continuous runtime field data. The AAS deployment processes entail three steps, as shown in Figure 18:

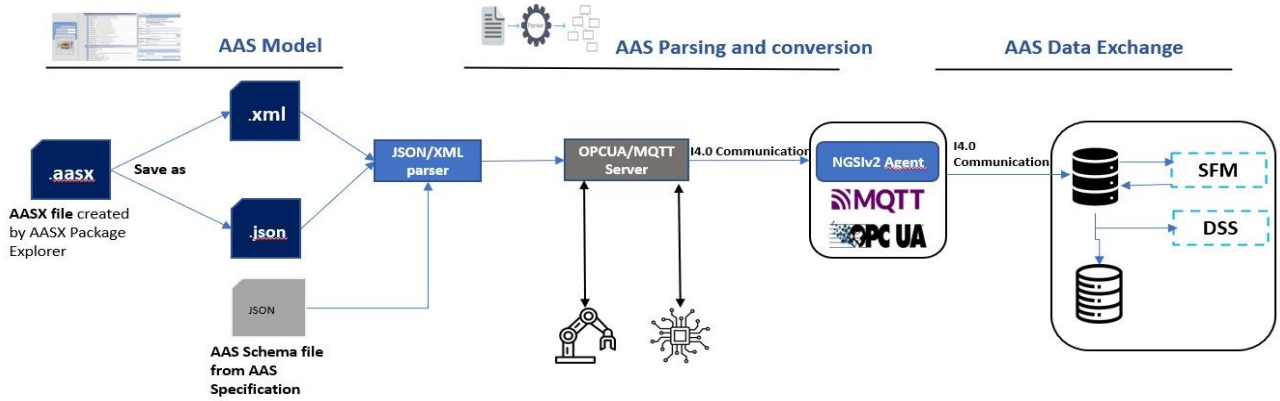


Figure 18 Implementation steps for AAS deployment – Source HSEL based on [22]

In step 1, the AASX package explorer is used to model the field assets, their properties and hierarchical relationships in an AASX file, which will also be saved as a JSON and XML files. Resulting from part 1 of the AAS specification, an AAS schema file (JSON format) that defines the elements for the AAS meta-model is also prepared.

In step 2, the interface file is used by the asset I4.0 gateway server to update static and real time values in the designated fields. The output of this step is an information metamodel containing context information.

In step 3, common interface information model is used by the components in MES for different functionalities such as data aggregation, data storage, process orchestration and decision making.

3.4. CoRoSect's System Architecture

CoRoSect's System Architecture [Deliverable 2.3] [Deliverable 2.4] was developed keeping in mind the DoW objective which is to create a novel digitalised integrated business solution based on RAMI 4.0. Based on the Industry 4.0 standard reference architecture model, the CoRoSect architecture considers merging of Shop floor assets and infrastructures with IoT technologies. The architecture development was modelled based on the requirements

In order for the system architecture to be compliable to Industry 4.0 reference architecture, interoperability and flexibility of CPS and components is needed. One of the key principles is to choose an architecture that covers the specific use-case requirements. In this scenario, it is required that the assets at the shop-floor level (OT) should be able to share data and information among themselves and also with the enterprise (IT) levels in an SoA way. The convergence of the OT and IT systems demands a specific middleware-like entity that can fulfil the service-oriented architecture requirement. In this regard, PERFoRM architecture [23] fits the CoRoSect infrastructure.

Figure 14 depicts the entities/components that are needed to realize the CoRoSect architecture. ***HSEL*** recommended the specific technologies, methodologies, interfaces, data modelling, and data routing (information flow mechanisms) that are required for process and service orchestrations.

The efficiency and flexibility can be improved by Implementing the I4.0 industrial automation framework. This can help exposing the functionalities of the insect farm's assets and enabling effective communication and integration of information. In an I4.0-compliant insect farm, the Manufacturing Execution System (MES) would act as the central hub for collecting, organizing, and distributing data from the shop floor while orchestrating the processes.

The MES is connected to Basyx [13], an open-source platform for industrial automation, which will enable the communication with FIWARE [12], a cloud-based platform for building IoT applications. FIWARE [12], in turn is integrated with NGSI, a standard for exchanging and representing data in IoT systems.

A high-level abstraction of data flows combined with standard technologies and protocols to control the farm process is detailed in Deliverable 2.4.

The architecture enables data flow from Assets to MES through a context broker, sometimes in scenarios when real time control is required the assets interface communicates with the SFM directly. Critical functions and controls need to be defined in such a way that the system is independent of real time constraints to ensure safety of the assets and the humans as well.

3.5. Standard Protocols (MQTT, OPCUA) and interfaces (API's)

The DIN SPEC 91345 RAMI4.0 specifies MQTT, OPC UA and Webservices (Figure 19) as the technologies/protocols that are compliant as they support service-oriented architecture [1].

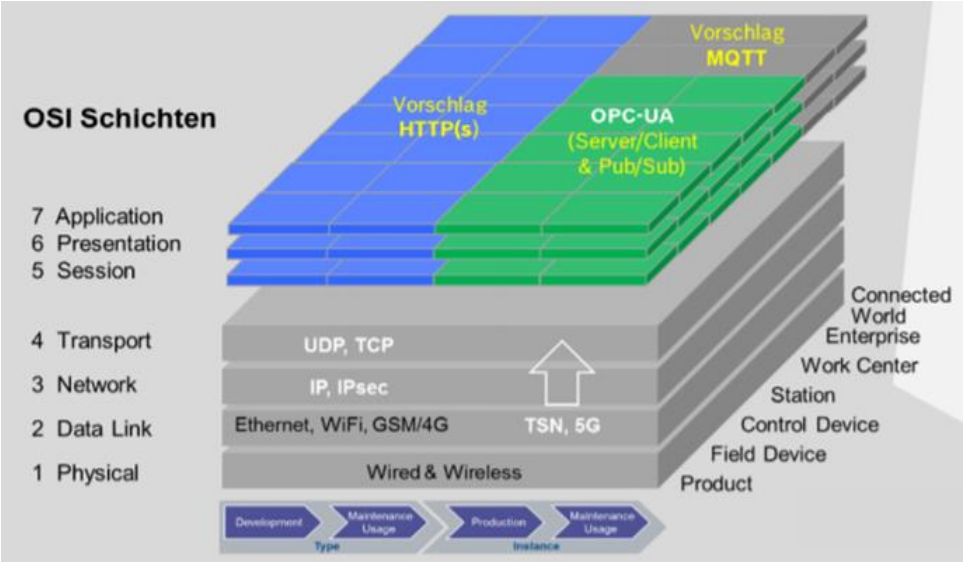


Figure 19 RAMI 4.0 compliant Communication protocols – Source [3]

MQTT and OPC UA are the most widely used industrial communication protocols. Depending upon the real-time communication constraints the protocols are chosen by the asset providers. In applications where real-time is a constraint, OPC UA is best suited as control and manipulation of assets is critical for the their safety and also the humans working with them (D-Cell, M-Cell/VI, Augmented Reality glasses (Hololens) [#2 - CERTH]). MQTT on the other hand is typically used when we need data from constrained devices such as sensors (I-CRATES, AGV, Object detection and Route Manager).

As shown in Figure 20, MQTT is divided into two parts: MQTT client and MQTT broker. In this implementation the assets publish data to the MQTT broker (MQTT Explorer) as topics (Sub-Models) and these topics contain the payload Sub-Models as JSON). The information is published by the client when the value of the topic changes by following the report by exception mechanism. The JSON payload is used as server interface by the MES (IMS), the context information from the asset is received by the context broker.

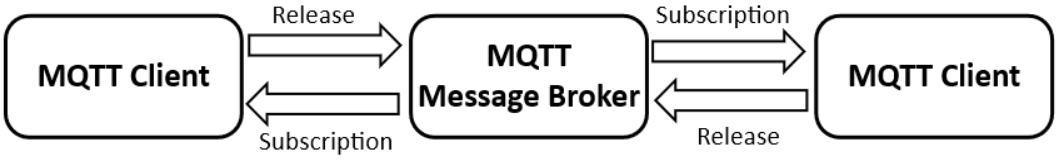


Figure 20 MQTT structure diagram – Source based on [24]

OPC UA on the other hand is widely used industrial protocol as it supports information modelling anddata transport over web services. The protocol is used by the cyber-physical world to connect to any assets in the IT system. The service interface (AAS) is converted to XML. The OPC UA server developed by asset provider is connected to the asset controller (PLC/ROS/SW) via a proprietary protocol. The context information of the asset is then published to MES (IMS) via I4.0 service gateway. In Figure 21 the general OPC UA Server - client architecture is shown

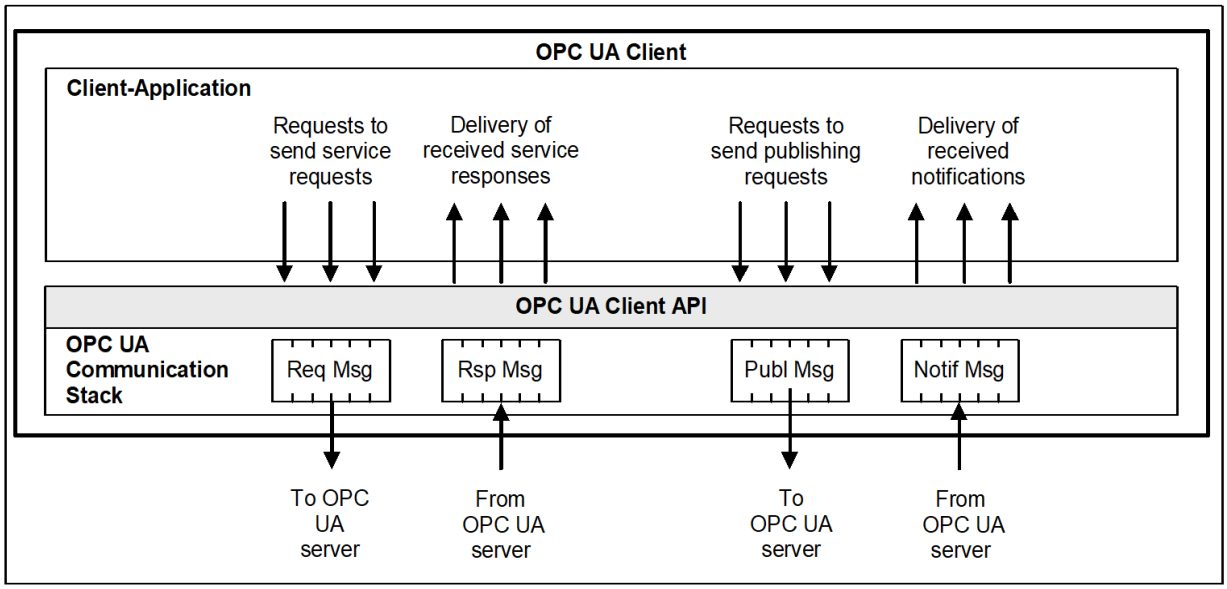


Figure 21 OPC UA Server - Client architecture - Source [25]

The simplest way for a Client and Server to exchange data is using the Read and Write OPC UA services, which allow an OPC UA Client to read and write one or more attributes of Nodes, maintained into the Address Space of the OPC UA Server; like most other services, the Read and Write services are optimised for bulk read/write operations and not for reading/writing single values.

A different and more sophisticated way to access data is based on Subscriptions and Monitored Items; this is the preferred method for clients needing cyclic updates of variable values. Subscriptions and Monitored Items are put on the top of a Session level, which is an intermediate between an OPC UA Client and an OPC UA Server created in the context of a Secure Channel [25].

4. Interfaces between Cyber physical systems and SFM

As detailed in Section 3.3.1, interfaces are required for the assets to interact with the MES (IMS) using standard protocols. These interfaces are defined and provided to each shop floor component providers. After exhaustive analysis of the functions and attributes associated to each function, we have formulated an excel sheet, this sheet [Annex 7.1] outlays the contents of AAS, each Shop floor component provider had proposed what information is needed by the SFM to orchestrate the farm processes. For example: **AGVPosition**, **MRobotStop**, **ICrateTemperature**based on common understanding, is a document containing relevant information about specific assets, their Sub-Models, type of elements, datatypes and other semantics that were created for each asset.This document is the reference for AAS implementation. Figure 22 illustrates an example of one asset document from the shop floor.

Asset	Description	Element type	Data type					
MRobotVI	MRobot and VI asset		string					
Submodel								
SM OperationalCapabilities	Description	Element type	Operation Variables Input	Operation Variables Input description	Input variable Data type	Operation Variables Output	Operation Variables output description	Output variable Data type
MRobotExecuteTask	To make MRobot execute a predefined task	Operation	taskID	To give the task id to be executed	int	CommandStatus	To let the the service consumer know that the command has been received and will be processed	boolean
MRobotStop	Stop execution of current tasks/motions	Operation	-	-	-	CommandStatus	To let the the service consumer know that the task has been stop	boolean
MRobotResume	Continue task after stop (if possible)	Operation	-	-	-	CommandStatus	To let the the service consumer know that the command has been received and will be processed	boolean
MRobotReturnErrorStack	Return exceptions trace	Operation	-	-	-	ExceptionMessage	To let the service consumer know about the exception messages	string
MRobotEmergencyStop	Emergency T1 stop	Operation	-	-	-	CommandStatus	To let the the service consumer know that the task has been stop by emergency	boolean

Figure 22 Reference excel document for each asset – Example Image – Source Screenshot

There are several methods to implement an AAS, one such is specified by Platform Industry 4.0, a package file format .AASX that can be generated using the tool described in [26], using the standard functions and properties of each asset. Their respective Asset Administration Shells are created using the Package explorer. Based on the open packaging conventions for representing an AAS [3], designed to be in compliance with the AAS meta-model, the AASX file encapsulates the AAS information including AAS model elements, properties, and additional files of any format in a structural way. Within the scope of current developments in projects and few industries, AASX is a widely acceptable data representation and exchange format for AASs.

The AASX package explorer tool [26] is a C#-based AAS editor that provides a series of functions to manipulate the AAS information. It also implements several APIs including representational state transfer (REST), MQTT, and open platform communications united architecture (OPC UA) for other software applications to access the AAS information, while security mechanisms, e.g., token-based authentication, are also embedded. Figure 23 shows the AASX development interface.

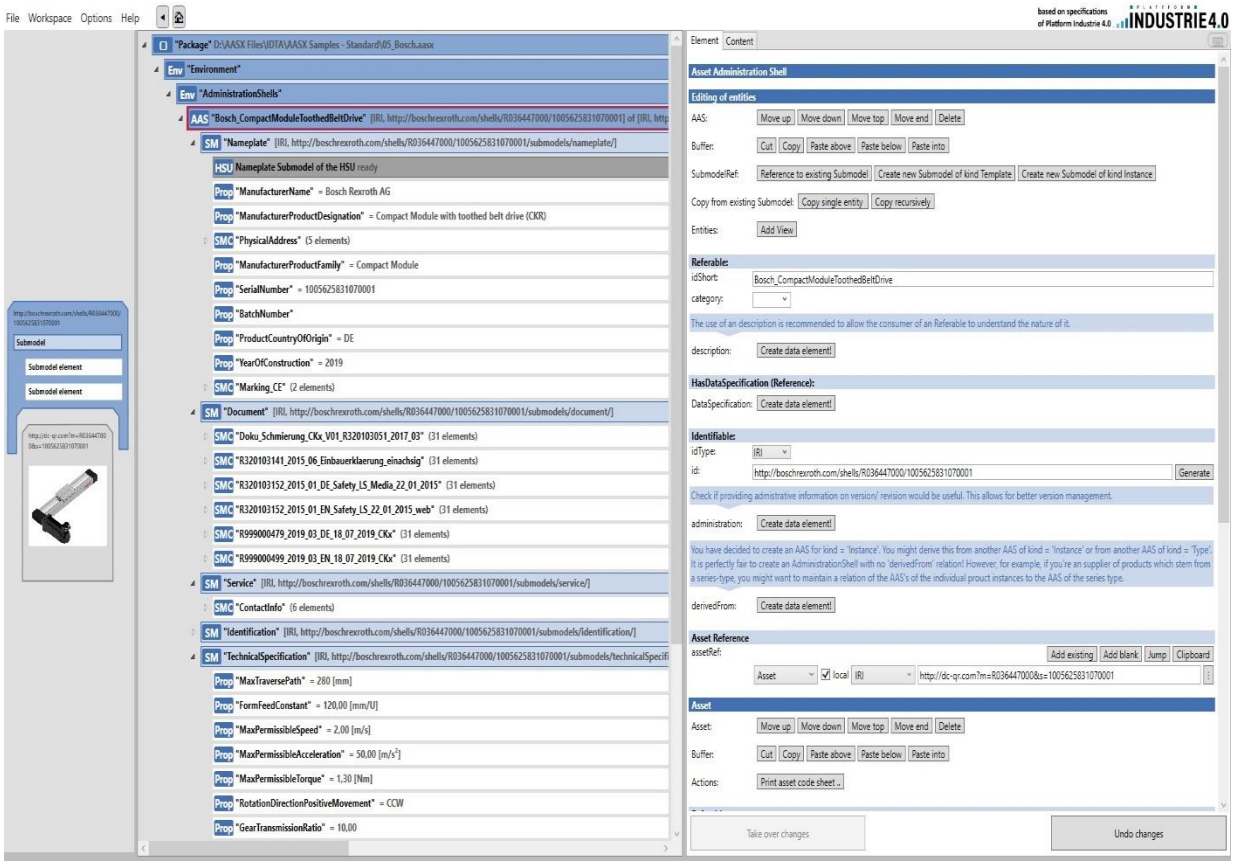


Figure 23 AASX Package Explorer UI – Example image – Source Screenshot

Also, based on the Reference implementation of I4.0 service gateway (3.2) the asset providers build servers (OPC UA, MQTT) on their premises. Technically, these servers can be built on top of the assets for functionality purposes or these servers can be built as an instance in cloud/ in local servers to bridge the communication between the assets and shop floor manager. As this deliverable only defines the interfaces, a brief overview of the interfaces is provided here. The detailed of integration are in the scope of Deliverable 9.2.

Assets provide real time information and the IMS captures this data using interface APIs (Basyx API [13]), the status/values of the asset and its actions are formatted as data according to the information model of the corresponding device and this data is sent as JSON payload to the IMS. The infrastructure that is built around each asset to collect, process and send this information is different for different assets as these assets range from Hardware sensors to software MES, but they follow a common methodology.

Methodology: All assets with AAS in the shop floor have their own I4.0 gateways running on either MQTT/OPC UA protocols, the service interface (AAS) exposes information about the assets, Sub-Models, Sub-Model elements to IMS via different entry points for different protocols and this entry point will also be used to send commands to the assets. The asset information is provided Sub-Models and the information inside Sub-Models will be used for dynamically updating the data to the IMS synchronously or asynchronously. The relevant context information will be implemented in decision making and orchestrating the farm processes by the MES.

The core of IMS implementation is the Basyx API [13], a context broker aggregates all the information from different connectors receiving data from different shop floor assets. When the SFM invokes a request based on DSS and the responses it receives from the assets, a request/query is sent via a API

GET method, a payload containing the request and InOut arguments is mapped against the AAS in the NGSI and a corresponding command is sent to the asset controller via the I4.0 gateway.

As discussed, each asset has its own set of functionalities and properties, which are specifically defined in the sections of each asset provider below with a brief description of asset functionalities.

A generic interface communication implementation scheme is shown in the Figure 24 below. For every asset, specific interface implementation is detailed in Deliverable 9.2.

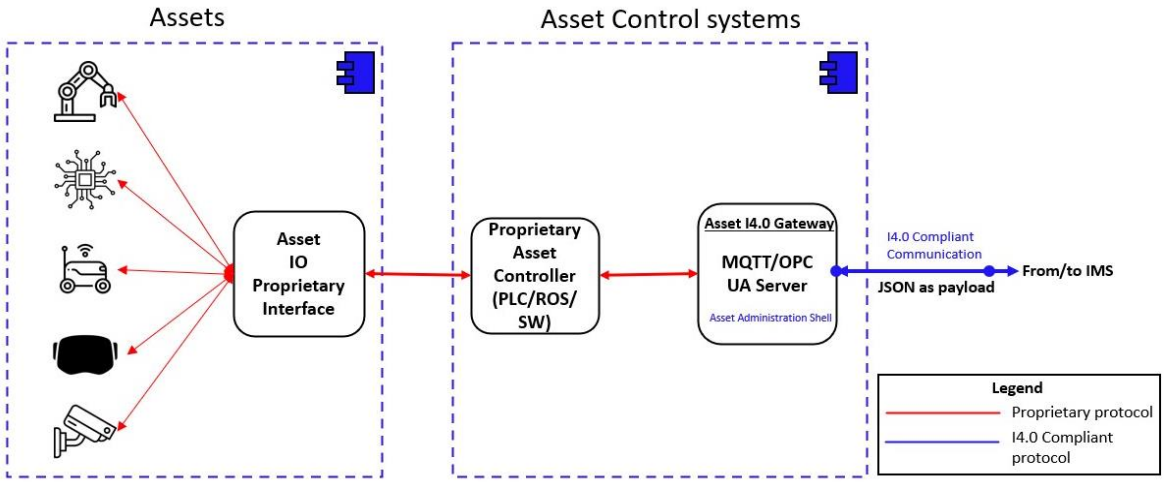


Figure 24 CoRoSect Interfaces implementation communication scheme.

4.1.1. Identified Assets

The assets listed in the following sections have been previously defined in the deliverable 2.3, section 4.4.2 Identification of Assets (In order to show which assets need to be digitized and to create their interfaces).

4.1.2. Stacking/De-staking Robot (D-Robot) Interface data

The D-Robot's main **functionality** is divided into two main procedures:

- **De-stacking procedure:** D-Robot picks/box crate from input pallet and places it in operation table.
- **Stacking procedure:** D-Robot picks crate/box form operation table and places it in output pallet.

The table summarizing the asset data is shown below:

Submodel Nameplate

Field	Description	datatype
ManufacturerName	Legally valid designation of the natural or judicial person which is directly responsible for the design, production, packaging and labelling of a product in respect to its being brought into circulation	string
ManufacturerProductDesignation	Short description of the product	string
Country code	Agreed upon symbol for unambiguous identification of a country	string

Street	Street name and house number	string
Zip	ZIP code of address	string
CityTown	Town or city of the company	string
StateCounty	State/county	string
ManufacturerProductFamily	2nd level of a 3-level manufacturer specific product hierarchy	string
YearOfConstruction	Year as completion date of object	string
SerialNumber	Unique combination of numbers and letters used to identify the device once it has been manufactured	string
ClassificationSystem	Classification System	string
DateOfManufacture	Date from which the production and / or development process is completed or from which a service is provided completely	string
ProductCountryOfOrigin	Country in which the product is manufactured (manufacturer country)	string
QrCode	QrCode	string
ProductIdentifier	ProductIdentifier	string

Submodel Technical data

Field	description	datatype
RobotMovementStatusConfigured	To let service consumer know about the movement status that are configured for the D-Robot	string *{0:"Moving", 1:"Stopped", 2:"EmergencyStopped"}*
StatusConfigured	To let service consumer know about the status that are configured for the D-Robot	string *{0:"Off", 1:"On", 2:"Under maintenance", 3:"Faulty", 4:"Out of service", 5:"Ready"}*
TaskStatusConfigured	To let service consumer know about the task status that are configured for the D-Robot	string *{0:"Paused", 1:"Executing", 2:"Success", 3:"Failed"}*
GeneralTasksConfigured	To let service consumer know about the general tasks that this D-Robot can perform and its associated task id's	string *{0:"move to x,y,z", 1: "Return to home position", …}*
SpecificTasksConfigured	To let service consumer know about the specific tasks that this D-Robot can perform and its associated task id's	string *{100:"D-Stack", 101: "Stack"}*

Submodel AssetConditionMonitoring

Field	description	datatype
RobotMovementStatus	Check D-Robot motion -> currently moving, stopped or emergency stopped	int (0-2)
Status	Check D-Robot overall status	int (0-5)
GripperObjectHeld	Check gripper status -> object is currently held	boolean
TaskStatus	Check status of the current task	int (0-3)
TaskID	To know for which task id the task status corresponds	int (0-… + 100-101)
DRobotCobotWorkspaceFree	Check that the shared workspace between D-robot and M-robot is free	boolean
DRobotAGVSharedWorkspaceFree	Check that the shared workspace between D-robot and AGV is free	boolean

Submodel OperationalCapabilities

Field	description	Input variable datatype
ExecuteGeneralTask	Make D-Robot execute a predefined task	int
DeStackCrate	Make D-Robot execute de-stacking of a crate into table	String ((x, y, z) crate position in JSON format)
StackCrate	Make D-Robot execute stacking of a crate into pallet	String ((x, y, z) pallet position in JSON format)
StopOperation	Stop execution of current task	-
ResumeOperation	Continue task after stop (if possible)	-
ReturnErrorStack	Return exceptions trace	-
EmergencyStop	Execute emergency T1 stop	-
InitializeCell	Initializes the cell status before a new pallet of crates enters in the cell	String ((x, y, z) cell configuration in JSON format)

4.1.3. Manipulation Robot including Visual Inspection (M-Robot/VI)

The M-Robot including the Visual Inspection Sensors is a core component in the CoRoSect Robotic Cell. The M-Robot's main objective is

- Manipulation of insects (e.g., picking, placing, sorting).

- Feeding of insects (e.g., adding feed to insect crates)
- AI-based visual inspection and monitoring of insects (e.g., visual monitoring of growth)
- Material handling for manipulating the insects' environments (e.g., adding/removing support structures into/from crates). The M-Robot is designed to collaborate with human co-workers. To fulfil its tasks the M-Robot is equipped with sensors and robust controllers to perform its tasks in a safe way.
  Based on these functionalities, the table below specifies the data associated to the asset that can be exchanged for achieving the objectives.

Submodel Nameplate

Field	description	datatype
ManufacturerName	Legally valid designation of the natural or judicial person which is directly responsible for the design, production, packaging and labeling of a product in respect to its being brought into circulation	string
ManufacturerProductDesignation	Short description of the product (short text)	string
CountryCode	Agreed upon symbol for unambiguous identification of a country	string
Street	Street name and house number	string
Zip	ZIP code of address	string
CityTown	Town or city of the company	string
StateCounty	State/country	string
ManufacturerProductFamily	2nd level of a 3 level manufacturer specific product hierarchy	string
YearOfConstruction	Year as completion date of object	string
SerialNumber	Unique combination of numbers and letters used to identify the device once it has been manufactured	string
ClassificationSystem	Classification System	string
DateOfManufacture	Date from which the production and / or development process is completed or from which a service is provided completely	string

ProductCountryOfOrigin	Country in which the product is manufactured (manufacturer country)	string
QrCode	QrCode	string
ProductIdentifier	ProductIdentifier	string

Submodel TechnicalData

Field	description	datatype
MRobotTaskConfigured	To let service consumer know about the tasks that this M-Robot can perform and its associated task ids	string *[{1: "Return to home position", 2:"Replace sensors", 3: "Perform quality management", 4: "Fill crate", 5: "Empty crate", 6: "Replace food and drink", 7: "Push crate", 8:"Add new beetles", 9:"Add wet feed", 10:"Pick up dead flies and place them into container", 11:"Place oviposition boards with hatched eggs into container", 12:"Place oviposition boards with fresh eggs into container", 13:"Install clean oviposition boards", 14:"Position the oviposition boards with fresh eggs into maturation racks"}]*
MRobotMovementStatusConfigured		string *[{0:"Moving", 1:"Stopped", 2:"EmergencyStopped"}]*
VIConfiguredInspectionType	To let service consumer know about the configured inspection type for the VI system	string
VIConfiguredFarmType	To let service consumer know about the configured farm type for the VI system	string
VIConfiguredDOL	to let service consumer know about the configured DOL type for the VI system	string
StatusConfigured	To let service consumer know about the status that are configured for the VI system	string
TaskStatusConfigured	To let service consumer know about the task status	string

Submodel AssetConditionMonitoring

Field	description	datatype

MRobotMovementStatus	To know if the robot is currently moving, stopped or emergency stopped	int
MRobotStatus	Telling about the overall status of the robot	int
MRobotObjectHeld	Check whether an object is currently held	boolean
MRobotTaskStatus	status of the given task to the robot	Int
MRobotTaskID	To know for which task id the task status corresponds	int
VITaskStatus	status of the given task to the VI	int
VITaskID	To know for which task id the task status corresponds	int
VIStatus	Telling about the overall status of the VI system	int
CobotDRobotWorkspaceFree	To tell that the shared workspace between M-robot & D-Robot is free	boolean

Submodel OperationalData

Field	description	datatype
VIResults	Sends all the result of visual inspection with order id as part of result	string

Submodel OperationalCapabilities

Field	description	datatype
MRobotExecuteTask	To make MRobot execute a predefined task	int
MRobotStop	Stop execution of current tasks/motions	boolean
MRobotResume	Continue task after stop (if possible)	boolean
MRobotReturnErrorStack	Return exceptions trace	string
MRobotEmergencyStop	Emergency T1 stop	boolean
VIStart	MES requests vision system to start visual inspection of a crate	Int / int / string/ int / datetime
VIStop	Immediately stop all currently running tasks	

VIReturnErrorStack	Return exceptions trace	
VIHistData	to return the recorded historical analysis data	String / int / datetime / datetime / int

4.1.4. Intelligent Crates (I-Crates)

I-Crates contain embedded environmental sensors and a processing and control node which together form so called Intelligent Integrated Sensors. The main functionality of I-Crate is to send the measured parameters inside the I-Crate to the IMS.

Information sent from the I-Crates to the shop floor manager contains the following elements:

Submodel OperationalData

Field	description	datatype
TemperatureMeasure	Temperature of the sensor in celsius	float
TemperatureTimestamp	Timestamp of temperature in GMT	string
HumidityMeasure	Humidity from the sensor in percentage	float
HumidityTimestamp	Timestamp of humidity in GMT	string
CO2Measure	CO2 level from sensor in ppm	float
CO2Timestamp	Timestamp of CO2 in GMT	string
MoistureMeasure	Moisture level from sensor in percentage	float
MoistureMeasureTimeStamp	Timestamp of Moisture in GMT	string
NH3Measure	Ammonium level from sensor in ppm	float
NH3Timestamp	Timestamp of NH3 in GMT	string
PHMeasure	pH level from sensor as a number	float
PHTimestamp	Timestamp of pH in GMT	string
I-CratesLocation	Geolocation of I-Crates	string

Submodel BillOfMaterials

Field	description	datatype
TemperatureSensorSensorIdentifiers	Unique identifier (MAC as default) of the temperature-sensor. If the I-CRATES configuration does not include this sensor, the description is “none”	string
HumiditySensorSensorIdentifiers	Unique identifier (MAC as default) of the humidity-sensor. If the I-CRATES configuration does not include this sensor, the description is “none”	string
CO2SensorSensorIdentifiers	Unique identifier (MAC as default) of the CO2-sensor. If the I-CRATES	string

	configuration does not include this sensor, the description is “none”	
NH3SensorSensorIdentifiers	Unique identifier (MAC as default) of the NH3-sensor. If the I-CRATES configuration does not include this sensor, the description is “none”	string
PHSensorSensorIdentifiers	Unique identifier (MAC as default) of the ph-sensor. If the I-CRATES configuration does not include this sensor, the description is “none”	string
MoistureSensorSensorIdentifiers	Unique identifier (MAC as default) of the Moisture sensor. If the I-CRATES configuration does not include this data, the description is “none”	string (long enough...)

4.1.5. Automated Guided Vehicle (AGV)

The Automated Guided Vehicle (AGV) is responsible for transporting the crates from and to the different robot cells on a robust and safe way.

The AGV communicates with the route manager its movement co-ordinates. The table below shows the data of the AGV:

Submodel OperationaCapability

Field	description	datatype
StartPause	Activates the pause mode.	-
StopPause	Deactivates the pause mode.	-
StartCharging	Activates the charging process.	-
StopCharging	Deactivates the charging process to send a new order	-
InitPosition	Resets (overrides) the pose of the AGV with the given parameters.	Array of Float and string
StateRequest	Requests the AGV to send a new state report	-
LogReport	Request the AGV to generate and store a log report.	string
Pick	Request the AGV to pick a load	Array of Float and string
Drop	Request the AGV to drop a load	Array of Float and string
DetectObject	AGV detects object (e.g. load, charging spot, free parking position).	string

FinePositioning	On a node, AGV will position exactly on a target	string
WaitForTrigger	AGV has to wait for a trigger on the AGV (e.g. button press, manual loading).	string
CancelOrder	AGV stops as soon as possible.	string

Submodel OperationalData

Field	description	datatype
OrderId	Unique order identification of the current order or the previous finished order. The orderId is kept until a new order is received.	string
OrderUpdateId	Order Update Identification to identify that an order update has been accepted by the AGV.	int
ZoneSetId	Unique ID of the zone set that the AGV currently uses for path planning. Must be the same as the one used in the order, otherwise the AGV has to reject the order.	string
LastNodeId	nodeID of last reached node or, if AGV is currently on a node, current node	string
LastNodeSequenceId	sequenceId of the last reached node or, if the AGV is currently on a node, sequenceId of current node	int
NodeStates	Array of nodeState-Objects that need to be traversed for fulfilling the order	array
NodeId	Unique node identification	string
NodeSequenceId	sequenceId to discern multiple nodes with same nodeId.	int
NodeDescription	Additional information on the node	string
NodePosition	Node position. The object is defined in ORDER topic Optional: master control has this information. Can be sent additionally, e. g. for debugging purposes.	JSON-Object
X	X-position on the map in reference to the map coordinate system. Precision is up to the specific implementation	float64
Y	Y-position on the map in reference to the map coordinate system. Precision is up to the specific implementation	float64
Theta	Orientation of the AGV on the node. Optional: vehicle can plan the path by itself. If defined, the AGV has to assume the theta angle on this node. If previous edge disallows rotation, the AGV must rotate on the node. If following edge has a differing orientation defined but disallows rotation, the AGV is to rotate on the node to the edges desired rotation before entering the edge	float64

AllowedDeviationXY	Orientation of the AGV on the node. Optional: vehicle can plan the path by itself. If defined, the AGV has to assume the theta angle on this node. If previous edge disallows rotation, the AGV must rotate on the node. If following edge has a differing orientation defined but disallows rotation, the AGV is to rotate on the node to the edges desired rotation before entering the edge	float64
AllowedDeviationTheta	Indicates how big the deviation of theta angle can be. The lowest acceptable angle is theta - allowedDevaitionTheta and the highest acceptable angle is theta + allowedDeviationTheta	float64
Nodereleased	Indicates if this node approached or approaching. “true” indicates that the node is part of the base. “false” indicates that the node is part of the horizon.	bool
EdgeStates	Array of edgeState-Objects that need to be traversed for fulfilling the order	array
EdgeId	Unique edge identification	string
EdgesequenceId	sequenceId to differentiate between multiple edges with the same edgeId	int
EdgeDescription	Additional information on the edge	string
Edgereleased	Indicates if this edge is processed or processing. “true” indicates that the edge is part of the base. “false” indicates that the edge is part of the horizon	bool
Trajectory	Trajectory of the path	JSON-Object
Degree	Defines the number of control points that influence any given point on the curve. Increasing the degree increases continuity.	float64
KnotVector	Sequence of parameter values that determines where and how the control points affect the NURBS curve. knotVector has size of number of control points + degree + 1.	array
ControlPoints	List of JSON controlPoint objects defining the control points of the NURBS, which includes the beginning and end point	array
AgvPosition	Current position of the AGV on the map.	JSON-object
PositionInitialized	“false”: position is not initialized “true”: position is initialized	bool
LocalizationScore	Describes the quality of the localization and therefore, can be used e. g. by SLAM AGVs to describe how accurate the current position information is.	float64
DeviationRange	Value for the deviation range of the position in meters.	float64

	Optional for vehicles that cannot estimate their deviation e.g. grid-based localization Only for logging and visualization purposes.	
MapId	Unique identification of the map in which the position is referenced. Each map has the same origin of coordinates. When an AGV uses an elevator, e.g., leading from a departure floor to a target floor, it will disappear off the map of the departure floor and spawn in the related lift node on the map of the target floor	string
MapDesc	Additional information on the map.	string
Velocity	The AGVs velocity in vehicle coordinates	JSON-Object
Vx	The AGVs velocity in its x direction	float64
Vy	The AGVs velocity in its y direction	float64
Omega	The AGVs turning speed around its z axis	float64
Loads	Loads that are currently handled by the AGV.	array
LoadId	Unique identification number of the load	string
LoadType	Type of load	string
LoadPosition	Indicates which load handling/carrying unit of the AGV is used,	string
BoundingBoxReference	Point of reference for the location of the bounding box. The point of reference is always the center of the bounding box's bottom surface (at height = 0) and is described in coordinates of the AGV's coordinate system.	JSON-Object
LoadDimensions	Dimensions of the load's bounding box in meters.	JSON-Object
Length	Absolute length of the load's bounding box.	float64
Width	Absolute width of the load's bounding box.	float64
Height	Absolute height of the load's bounding box.	float64
Weight	Absolute weight of the load measured in KG	int
NewBaseRequest	"true": AGV is almost at the end of the base and will reduce speed if no new base is transmitted. Trigger for master control to send a new base. "false": no base update required.	bool
DistanceSinceLastNode	Used by line guided vehicles to indicate the distance it has been driving past	int
ActionStates	Contains a list of the current actions and the actions which are yet to be finished. This may include actions from previous nodes that are still in progress. When an action is completed, an updated state message is published with actionStatus set to finished and if applicable with	array

	the corresponding resultDescription. The action state is kept until a new order is received	
Actionid	-	string
ActionType	actionType of the action	string
ActionDescription	Additional information on the current action	string
ActionStatus	See Action sheet for process in each action	int
ResultDescription	Description of the result if necessary e.g., the result of a RFID-read, BarcodeScanner, Measured Weight	string

Submodel AssetConditionMonitoring

Field	description	datatype
Status	Telling about the overall status of the agv	int
ConnectionState	Connection status of AGV for other assets, managed by AGV for ONLINE, OFFLINE, Broker can set the state to CONNECTIONBROKEN when connection is lost	int
BatteryState	Contains all battery-related information.	JSON-Object
BatteryCharge	State of Charge: if AGV only provides values for good or bad battery levels, these will be indicated	float64
BatteryVoltage	BatteryVoltage	float64
BatteryHealth	State of health	int8
Charging	"true": charging in progress "false": AGV is currently not charging	bool
Reach	Estimated reach with current State of Charge	int
OperatingMode	Current Mode of the vehicle	int
Errors	Array of error-objects. All active errors of the AGV should be in the list. An empty array indicates that the AGV has no active errors.	array
ErrorType	Type/Name of error	string
ErrorReferences	Array of references to identify the source of the error (e. g. headerId, orderId, actionId, … and its values)	array
Driving	"true": indicates that the AGV is driving and/or rotating. Other movements of the AGV (e.g., lift movements) are not included here. "false": indicates that the AGV is neither driving nor rotating	bool
Paused	True: AGV is currently in a paused state, either because of the push of a physical button on the AGV or because of an instantAction. The AGV can	bool

	resume the order. False: The AGV is currently not in a paused state.	
SafetyState	Contains all safety-related information	array
Estop	Enum {autoAck, manual, remote, none} Acknowledge-Type of eStop: autoAck: auto-acknowledgeable e-stop is activated e.g., by bumper or protective field manual: e-stop has to be acknowledged manually at the vehicle remote: facility e-stop has to be acknowledged remotely none: no e-stop activated	int
FieldViolation	Protective field violation. "true": field is violated "false": field is not violated	bool

4.1.6. Augmented Reality glasses (Hololens)

Human Robot collaboration environment safety is one of the primary objectives and hence AR can be used as one of the tools to improve safety by providing visual cues and real-time information and feedback about robot actions.

Microsoft's HoloLens 2 is a see-through-based augmented reality mobile device that contains various sensors and is an essential component regarding the situation awareness of the user, during human-robot collaboration schemes.

The HoloLens 2 AAS holds information on the HoloLens's 2 capabilities, functionalities and properties. The table below shows the data associated with the Hololens 2:

Submodel NamePlate

Field	description	datatype
ManufacturerName	Circulation	string
ManufacturerProductDesignation	Short description of the product (short text)	string
CountryCode	Agreed upon symbol for unambiguous identification of a country	string
Street	Street name and house number	string
Zip	ZIP code of address	string
CityTown	Town or city of the company	string
StateCounty	State/country	string

ManufacturerProductFamily	2nd level of a 3 level manufacturer specific product hierarchy	string
YearOfConstruction	Year as completion date of object	string
SerialNumber	Unique combination of numbers and letters used to identify the device once it has been manufactured	string
ClassificationSystem	Classification System	string
DateOfManufacture	Date from which the production and / or development process is completed or from which a service is provided completely	string
ProductCountryOfOrigin	Country in which the product is manufactured (manufacturer country)	string
QrCode	QrCode	string
ProductIdentifier	ProductIdentifier	string

Submodel TechnicalData

Field	description	datatype
ConfiguredAssets	Contains a list of configured assets in HoloLens2	string
StatusConfiguration	To let service consumer know about the status that are configured for the HoloLens	string
MessagesConfiguration	To let service consumer know about the status that are configured for the HoloLens	string

Submodel AssetConditionMonitoring

Field	description	datatype
Status	Telling about the overall status of the HoloLens	int
DisplayStopped	The function is used by any service consumer to display any kind of message on HoloLens	boolean

Submodel OperationalCapabilities

Field	description	datatype
DisplayMessage	The function is used by any service consumer to display any kind of message on HoloLens	MessageID / AssetID

DisplayTrajectory	The function is used by any service consumer to display the trajectory it has planned to be displayed on the HoloLens	Trajectory / AssetID
StopDisplayMessage	The function is used by any service consumer to display any kind of message on HoloLens	AssetID
StopDisplayTrajectory	The function is used by any service consumer to display any kind of message on HoloLens	AssetID
ReturnsStackError	Return exceptions trace	

4.1.7. Route Manager

Route manager deals with the creation and tracking of the routes of the robots. Route Manager is a pure software module.

Route Manager communicates with three modules:

- Shoop Floor Manager
- AGV
- Object Detector

Submodel OperationalCapability

Field	description	datatype
NewRoute	to request the Route Manager for new route	string
CancelRoute	to request the Route Manager for cancelling the route request	String string
StartRoute	request to start a named route	

Submodel OperationalData

Field	description	datatype
MapPetitions	Sends the current elements in the map	string
RouteStatus	Status marking when a route has finished	string

Submodel AssetConditionMonitoring

Field	description	datatype
Status	Telling about the overall status of the route manager	int
ErrorMessage	Telling about the error message that has happened in route manager	string
InformationSourceforStatus	Telling about the information source for status	string
TimeofStatusChange	Telling about the time of status change	datetime

4.1.8. Object detector

Obstacles' detector deals with the detection of static or dynamic obstacles in the shop floor. A complete description of the Obstacles Detector can be found in *Deliverable 6.7 Safety concept for robotic systems (planning)*(M12) and especially in *6.8 Safety concept for robotic systems (creation)* due to M24.

Obstacles' Detector is a pure software component associated with one or many fixed IP cameras.

Table below shows the data of the Obstacle detector:
Submodel OperationalData

Field	description	datatype
InformationSource	Source of information of detection	string
TimeofDetection	Time of Detection	datetime
DetectionUnqiueId	Unique Message Id associated to the detection	string
TypeofObstacle	Real Asset Type of the object detected	string
DetectedObjectPosition	Latitude and longitude of the detected object	string
PredictedTrajectory	Latitude and longitude of the detected object in 0 sec, 1 sec and 2 sec	string
DetectionMessage	Some extra information with regards to detected object	string

5. Data Security

5.1. Data Consistency

The role of HSEL is also to ensure the correctness of the data and the structure of data the shop floor asset controllers send to the IMS and vice versa, a common MQTT broker and a OPC UA (Ua Expert) client is implemented where all assets can send their information to. As semantics and syntax play a key role in implementing AASs, the information published and subscribed by the assets need to be compliant as per ZVEI recommendations published in [3]. In an Industry 4.0 environment where multiple systems are involved and interactions between systems generate high volumes of data, sensitive information of the assets need to be protected from unauthorised accesses. The concept of security demands Confidentiality, Integrity and Availability of data (CIA Triad), in this respect all the context information exchanged between IMS and assets at shop floor is secured and encrypted. Details of CoRoSect data security are explained in Deliverable2.4 (Advanced System Architecture).

5.2. Total Integration

CoRoSect's interfaces are defined based on RAMI4.0 specifications, independent of shop floor protocols, a generic metamodel of information of each asset is defined and the metainformation model is deployed by all technology providers in their respective I4.0 gateways implemented at shop floor level. During the integration phase all the metamodels of information of all assets will be integrated for achieving different functionalities that collectively achieve the digitalisation of Insect farms. The SFM orchestrates the farm processes using the developed interfaces

6. Conclusion

According to what was presented in the introduction, this deliverable proposes the integration plan and concretely define the interfaces for the digitization of the assets in order to cover the requirements for the orchestration of services required for the particular use case.

Initially, the requirements for Interfaces and Integration were defined to create a compatible ecosystem, in which each asset can share functionalities that allow the orchestration from the MES to the Shop Floor or vice versa. Due to the multiplicity of Assets and in order to achieve a complete integration, different considerations had to be taken into account. On the one hand, there are different communication protocols (OPC UA, MQTT) and on the other hand, an integration with possible applications that may exist has to be carried out to ensure a coherent flow between the different hierarchical levels. In addition, part of the basis for the generation of this deliverable have been taken from Deliverable 2.3, since it laids the foundations on which the integration plan of the CoRoSect devices and systems was based.

In turn, the information of each relevant Asset has been structured and published under different points of view (different formats), through its AAS. For this reason, concrete information had to be generated according to the intended functions of each asset according to the use case. This was done in a standardized, specific way and using specific tools and resources.

With this, it is clear that in this deliverable it was necessary to focus on the different interfaces required to achieve the integration of the different assets in a coherent and compatible ecosystem with the guidelines of Industry 4.0 through the proposed architecture model RAMI 4.0.

The deliverable defines the methodology and approach plan to integrate the assets of CoRoSect shop floors to achieve interoperability between OT and IT levels. Firstly, the deliverable focused on detailing the steps of developing the I4.0 gateways that each technology provider will implement and the interfaces required by I4.0 gateways. These interfaces are needed at the OT level for exchanging information to the MES for process orchestration. And finally, the deliverable specifies the method of implementing these interfaces for smart mechatronics and IoT based devices using I4.0 compliant communication technologies and frameworks. The project follows a service-oriented architecture based on the OASIS RM SOA [4] and the I4.0 gateway and uses the Asset Administration Shell (AAS) to connect assets to the system. A FIWARE [12] context broker (Figure 13) was used to manage the interactions between the assets, and provides a service using a REST API server and NGSI interface. This allows for a coherent, scalable, and safe environment for insect farming, and automates repetitive and physically demanding tasks.

The interfaces will be developed using open-source technologies (FIWARE [12], Basyx [13], AASX Package Explorer [26]) according to the proposal of this deliverable. These are developed based on a reference schema resulting from several standard implementation references. On the other hand, a common interface had to be provided for asset-related data loading. For this purpose, a preformatted excel spreadsheet was provided to facilitate the loading of information and thus be able to correctly generate the different assets administration shells. Then, using specific tools [26], it was possible to generate the compatible data schemas according to the ones proposed by the I4.0 platform in JSON and XML.

Finally, the outcomes of this deliverable were: the interfaces, the specific guidelines for implementing these interfaces and the proposed technologies will be used by technical work packages (WP2, WP4, WP5, WP6, WP7, WP8 and WP9) to enable the assets to work in a coherent manner. Based on these

implementations the integration of all CoRoSect system components (OT and IT) will be detailed in Deliverable 9.2 (Integrated CoRoSect Platform).

7. Annex

7.1. Spreadsheets used to standardize the information load of the Asset Administration Shell

Links to the different folders.

- Stacking/De-staking Robot (D-Robot):
  - Link: AAS_DRobot.xlsx
- Manipulation Robot including Visual Inspection (M-Robot/VI)
  - Link: AAS_MRobot_VI.xlsx
- Intelligent Crates (I-Crates)
  - Link: AASIC.xlsx
- Automated Guided Vehicle (AGV)
  - Link: AAS AGV.xlsx
- CoRoSect Partner N°2 CERTH Augmented Reality glasses (Hololens)
  - Link: AAS_Holoens.xlsx
- Route Manager
  - Link: AAS Route Manager.xlsx
- Object detector
  - Link: AAS ObjectDectector.xlsx

Note for external reviewers: Kindly send an email to CoRoSect project leader, Mr. Rico Möckel requesting the AAS Excel files.

Email: **rico.mockel@maastrichtuniversity.nl**

7.2. Standardized output formats of the Asset Administration Shell

- Stacking/De-staking Robot (D-Robot):
  - Link: AAS_DRobot.xml
- Manipulation Robot including Visual Inspection (M-Robot/VI)
  - Link: AAS_MRobot_VI.xml
- Intelligent Crates (I-Crates)
  - Link: ICrate JSON
- Automated Guided Vehicle (AGV)
  - Link: AGV JSON
- CoRoSect Partner N°2 CERTH Augmented Reality glasses (Hololens)
  - Link: AASHololens.xml
- Route Manager
  - Link: RM JSON_files
- Object detector
  - Link: Object Detector JSON_files

References

[1] DIN SPEC 91345. Reference Architecture Model Industrie 4.0 (RAMI4.0), Released: April 2016

[2] DIN EN 62890. Life-cycle-management for and products used in industrial-process measurement, control and automation, Released: May 2017.

[3] Industrie 4.0 Plattform, Industrial Digital Twin Association, and ZVEI. Details of the asset administration shell. Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC02). URL: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html. Last Access: 01.12.2022

[4] OASIS. Reference architecture foundation for service-oriented architecture version 1.0. URL: http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html. Last Access: 01.12.2022

[5] Colombo, Armando Walter. Lectures in Digitalization of Industrial Cyber Physical Systems. Master in industrial informatics Course, Hochschule Emden Leer (Germany), 2022.

[6] ANSI / ISA 95. Enterprise - Control System Integration.

[7] ANSI / ISA 88. Batch Control.

[8] DIN EN / IEC 62264. Life-cycle management for systems and products used in industrial-process measurement, control and automation. Released: April 2017.

[9] DIN EN / IEC 61512. Batch control systems. Released: August 1997.

[10] ZWEI. Ergebnispapier “Struktur der Verwaltungsschale - Fortentwicklung des Referenzmodells für die Industrie 4.0-Komponente”. URL: https://www.zvei.org/presse-medien/publikationen/struktur-der-verwaltungsschale/. Last Access: 05.12.2022

[11] ISO/TS 29002-5. Industrial automation systems and integration — Exchange of characteristic data — Part 5: Identification scheme. Released: February 2009.

[12] FIWARE Smart Solution, Release 8.3.0. URL: https://github.com/FIWARE/catalogue

[13] Eclispe Basyx Foundation – Middleware for Industry 4.0 https://basyx.CoRoSect.ari-aidata.eu/docs

[14] IEC 61511. Functional safety - Safety instrumented systems for the process industry sector.

[15] IEC 62061. Safety of machinery - Functional safety of safety-related control systems.

[16] IEC 62453: Field device tool (FDT) interface specification

[17] IEC 61804. Devices and intergration in enterprise systems - Function blocks (FB) for process control and electronic device description language (EDDL)

[18] Open source implementation of OPC UA. URL: https://github.com/OPCFoundation/UA-.NETStandard

[19] Pure Python OPC-UA Client and Server. URL: https://github.com/FreeOpcUa/python-opcua

[20] OPC Unified Architecture.NET Standard. URL: https://github.com/OPCFoundation/UA-.NETStandard

[21] UaExpert A full featured OPC UA client. URL: https://www.unified-automation.com/products/development-tools/uaexpert.html

[22] X. Ye, S. H. Hong, W. S. Song, Y. C. Kim and X. Zhang, "An Industry 4.0 Asset Administration Shell-Enabled Digital Solution for Robot-Based Manufacturing Systems," in *IEEE Access*, vol. 9, pp. 154448-154459, 2021, DoI: 10.1109/ACCESS.2021.3128580.

[23] Angione, Giacomo & Barbosa, José & Gosewehr, Frederik & Leitão, Paulo & Massa, Daniele & Matos, João & Peres, Ricardo & Rocha, Andre & Wermann, Jeffrey. (2017). Integration and Deployment of a Distributed and Pluggable Industrial Architecture for the PERFoRM Project. Procedia Manufacturing. 11. 896-904. 10.1016/j.promfg.2017.07.193.

[24] Zuoling Niu 2021 J. Phys. ” Research and Implementation of Internet of Things Communication System Based on MQTT Protocol” Conf. Ser. 2023 012019

[25] Salvatore Cavalieri, Ferdinando Chiacchio, “Analysis of OPC UA performances,” *Computer Standards & Interfaces,* vol. 36, no. 1, pp. 165-177, 2013.

[26] AASX Package Explorer tool. URL: https://github.com/admin-shell-io/aasx-package-explorer

CoRoSect

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016953