D9.4 - Test cases for the CoRoSect ecosystem

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)	ATOS
Contributor(s)	UM, CERTH, OAMK, TECNOVA, AGVR, ROB, HSEL
Work Package	WP9. Secure platform integration
Delivery Date (DoA)	2023-01-31
Actual Delivery Date	11/08/2023
Abstract:	This document summarises the test bed for the validation of each Shop Floor Component integration with the Shop Floor Manager. These tests rely on the first version of the integrated CoRoSect system and describe how the CoRoSect's Digital Twins are used to support data and commands' flows according to the RAMI4.0 interfaces defined in D9.1. On a second stage, the CoRoSect subsystems' orchestration (Shop floor plus MES) needed to run rearing processes is also tested. This complete baseline validation of the CoRoSect's integrated system enables it for the pilots' evaluation in WP10.

Document Revision History			
Date	**Version**	**Author/Contributor / Reviewer**	**Summary of main changes**
15/02/2023	V0.1	ATOS	First proposed ToC
15/05/2023	V1.0	ATOS	First round of contributions for DT tests.
25/06/2023	V1.3	ATOS/HSEL	Orchestration's tests added
04/07/2023	V1.4	ATOS	Updated tables' tests
07/07/2023	V1.5	ATOS	Internal review version
10/08/2023	V2.0	ATOS	Final version to be delivered

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 name**	**Country**
1	UNIVERSITEIT MAASTRICHT https://www.maastrichtuniversity.nl/	UM	NL
2	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS https://www.certh.gr/	CERTH	GR
3	HOCHSCHULE EMDEN/LEER https://www.hs-emden-leer.de/en/	HSEL	GER
4	LUONNONVARAKESKUS https://www.luke.fi/	LUKE	FIN
5	OULUN AMMATTIKORKEAKOULU OY - OULU UNIVERSITY OF APPLIED SCIENCES https://www.oamk.fi/fi/	OAMK	FIN
6	FUNDACION PARA LAS TECNOLOGIAS AUXILIARES DE LA AGRICULTURA http://www.fundaciontecnova.com/	TECNOVA	ES
7	KATHOLIEKE UNIVERSITEIT LEUVEN https://www.kuleuven.be/kuleuven/	KU LEUVEN	BEL
8	ATOS IT SOLUTIONS AND SERVICES IBERIA SL https://atos.net/en/	ATOS	ES
9	ROBOTNIK AUTOMATION SLL http://www.robotnik.es/	ROB	ES
10	AGVR BV www.agvegroup.com	AGVR	NL
11	NASEKOMO AD https://nasekomo.life/	NASEKOMO	BG
12	ENTOMOTECH SL http://entomotech.es/	ENTOMOTECH	ES
13	ENTOCYCLE LTD https://www.entocycle.com/	ENTOCYCLE	GB
14	SOCIETA AGRICOLA ITALIAN CRICKET FARM SRL https://www.italiancricketfarm.com/	ICF	IT
15	INVERTAPRO AS https://www.invertapro.com/	INVERTAPRO	NOR
16	FIELD LAB ROBOTICS BV https://www.fieldlabrobotics.com/	FLR	NL
17	FoodScale Hub https://foodscalehub.com/	FSH	RS
18	AgriFood Lithuania DIH https://www.agrifood.lt/	AFL	LT
19	CENTRO INTERNAZIONALE DI ALTISTUDI AGRONOMICI MEDITERRANEI http://www.iamb.it/	CIHEAM	IT

LEGAL NOTICE

Table of Contents

Executive Summary

This document is part of the CoRoSect's System Integration and Evaluation process, developed mainly in WP9 but relying on the System Architecture presented in Task 2.3 and the Service-Oriented Information Management System (IMS) developed within WP4 (Task 4.3). Its purpose is to describe the executed tests on the different CoRoSect system interfaces (Task 9.1), to validate all the data and commands' flows that enable the CoRoSect Integrated System version 1 (Task 9.2) to be used within the incoming pilots.

This is the first deliverable from Task 9.3 (Testing, optimization, and technical validation) and specifically, it shows the Digital Twins (DT) developed within WP9 to map the project's Shop Floor Components and demonstrates how these support the commands and data flows between the CoRoSect's Shop Floor Manager (SFM) and the CoRoSect's Shop Floor Components (and vice versa). These flows are evaluated using the IMS within the Integrated CoRoSect Platform Release version 1, presented in D9.2, plus the OPC servers instances provided by CoRoSect's OPC native robots and the integrated MQTT broker supporting the MQTT RAMI4.0 compliant communications, according to the developments shown in D9.2 and D2.3. A specific instance of the CoRoSect's Shop Floor Manager with its embedded Decision Support System on top of the testing scenario, triggers and controls data requests, commands, and events' management, by testing the synchronous (Query/Retrieve) and asynchronous (Subscriptions) channels.

Within the introduced framework, this text presents the results of the first round of integration tests for the full CoRoSect system that validates:

- Shop Floor components' interfaces evaluation, designed and implemented in WP9, used to upload data from robots and invoke supported commands. These are developed for:
    - OPC-UA components, including: MRobot plus Visual Inspection subsystem; DRobot; and HoloLens devices.
    - MQTT reporting devices: AGV; Route Manager; and Objects Detector
- IMS RAMI4.0 compliant interface (using Eclipse BaSyx implementation) to integrate with the Shop Floor Manager and the Decision Support System
- Shop Floor Manager orchestration of the Shop Floor Components, through the IMS
- The system general functionalities implementations, regarding the data provisioning and user interfaces to present shop floor data and shop floor management

All these together will enable the CoRoSect System to support the incoming pilots in insects' farms and provide the required feedback to improve its performance. This testing and evaluation process will continue during the pilots' execution to complete the task 9.3 objectives that will be presented in D9.5.

List of tables

List of figures

List of Abbreviations and Acronyms	
AAS	Asset Administration Shell
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
API	Application Programming Interface
BPMN	Business Process Model and Notation
DMS	Data Management System
D-Robot	(Stacking) De-Stacking Robot
DS	Decision-Making System
DSS	Decision Support System
DT	Digital Twin
DX.Y	Deliverable X.Y
ERP	Enterprise Resource Planning
HRC	Human-Robot collaboration
HTTP	Hypertext Transfer Protocol
ICF	Italian Cricket Farm
ID	Identification
IIoT	Industrial Internet of Things
IMS	Information Management System
IoT	Internet of Things
IT	Information Technology
JSON	JavaScript Object Notation
Mx	Month X
MES	Manufacturing Execution System
ML/DL	Machine Learning / Deep Learning
MQTT	Message Queuing Telemetry Transport
M-Robot	Manipulation Robot
NGSI	Next Generation Service Interfaces
OD	Objects' Detector
OPC-UA	OLE (Object Linking and Embedding) for Process Control-Unified Architecture
OT	Operations Technology
Q&R	Query & Retrieve
RAMI4.0	Reference Architecture Model for Industry 4.0
RCS	Robot Control System
REST	Representational State Transfer
RM	Route Manager
ROS	Robot Operating System
SFM	Shop Floor Manager
SLAM	Simultaneous Localization And Mapping
SQL	Structured Query Language
TCP	Transmission Control Protocol
VI	Visual Inspection
WP	Work Package

1 Introduction

This deliverable closes the first operational version of the CoRoSect integrated system by **testing and evaluating the communication channels, covering data and commands' flows between its components.** With this, the purpose of this document is twofold: i) on one side, it validates the communication protocols and flows driven by the CoRoSect's IMS to link the project's IT systems, (represented here by the Shop Floor Manager and the Decision Support System) with the OT layer that composes the CoRoSect's Shop Floor; whilst on the other side, ii) it ensures that the executed commands (and data responses) by each integrated system (mainly robots and devices deployed at the shop floor) match the expected behaviour. These two targets enable the **CoRoSect's system orchestration** among all its shop floor components, driven by its shop floor manager. The carried out validation process enables the CoRoSect Integrated System to jump to the next stage and support the use cases execution during the CoRoSect's pilots in the insects' farms.

1.1. Scope of the deliverable

The scope is to resume the **integrated system validation** process through a series of communication tests that are applied between the software components of the CoRoSect System (Version 1.0). A process that guarantees the proper performance of the required functionalities and enables the RAMI4.0 compliant shop floor data access and distribution. In the same way, the SFM must be able to identify, command and control all the project's shop floor layers, according, also, to RAMI4.0 guidelines. Aligned, this test bed involved both: a) the IMS, SFM, DSS and AAS developers (IT layer); with b) the devices and shop floor mechatronics providers. These lasts actors will also validate the proper physical execution of the corresponding received commands and the generated responses.

The test cases will include all commands (**operations**) and data sets (**properties**) reported by each component's interface. The interfaces have been developed within Task 9.1 and compose each corresponding I4.0 **Asset Administration Shell** (AAS) which, in turn, are implemented by its **Digital Twin** (DT) within the CoRoSect IMS.

This set of Digital Twins supports all the data and commands' flows as the core of the CoRoSect MES functionalities. The test cases will validate both, the integrated system and the Digital Twins approach defined to implement it. This **qualifies the developed system for use in CoRoSect pilots**. In this sense, it is worth to remark that not all the properties and commands initially identified in D9.1 interfaces have been fully implemented for this first version of the CoRoSect Integrated System, because they are not needed to perform CoRoSect's use cases. The focus of this validation process has been set on the properties and commands required to support all the use cases defined by WP10.

1.2. Relationships with other tasks and deliverables

As mentioned, the resumed test cases validate the CoRoSect Integrated System which is the result of the work done so far by different tasks within different project's Work Packages. Figure 1 represents these dependencies that make possible the work of the task 9.3 and its validation process.

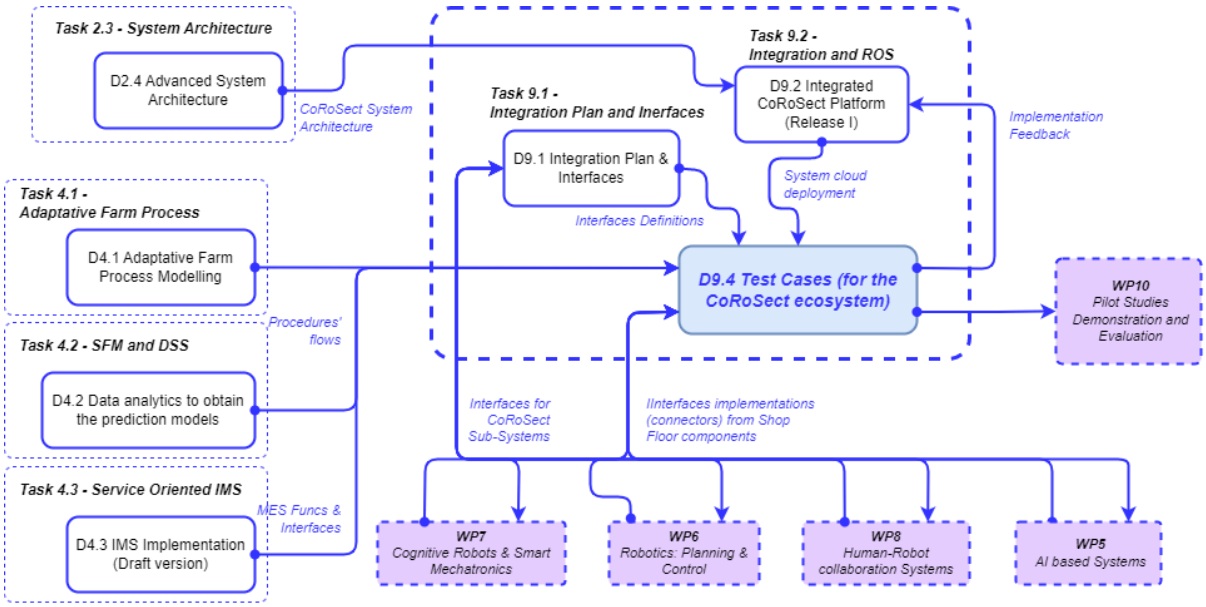

Figure 1. Test Cases for the CoRoSect ecosystem main dependences and outcomes

According to Figure 1 the dependencies and results linked to this deliverable are as follows:

- **From WP2** - Use-cases, user requirements and system architectures
  - Specifically, Task 2.3 (D2.4) provides the system architecture and the roles of each component that guide the implementation of the CoRoSect system to be validated. This is also the source of the functionalities that must be guaranteed.
- **From WP4** - Farm-level modelling and orchestration
  - Task 4.1 (D4.1) from a generic perspective, introduced the processes flows within the system's core that define the commands and data flows respectively. These must be aligned for the proper final CoRoSect's system specification.
  - Task 4.2 (D4.2) contributes to the processes definitions and functionalities required in terms of devices' control capabilities and required datasets to implement decision support capabilities.
  - Task 4.3 provides the design of the internal IMS functionalities, data sharing and commands interfaces plus the Digital Twins implementation. All these will be validated (and enhanced) during the validation test cases.
- **From WP9** - Secure platform integration (itself)
  - Task 9.1 (D9.1) defines the RAMI4.0 compliant interfaces (using I4.0 Asset Administration Shells) to communicate the IMS with the shop floor components at the southbound plus the corresponding interface (also RAMI4.0 compliant) at the northbound to link the SFM with the IMS, as well as other ERP systems in the farm. All these interfaces are to be validated in the test cases.
  - Task 9.2 (D9.2) contributes with the instance of the CoRoSect System (V1) that supports all the resumed test cases. In turn, as validation process feedback, this task will get improvements and detected bugs to enhance the final CoRoSect System version.
- **From technical Work Packages**
  - The rest of the technical work packages (WP5 to WP8) develops the different shop floor components and supports the physical validation of the sent commands plus the

proper dataset updates, using their corresponding interfaces (as RAMI4.0 compliant AAS)

- **To WP10** - Pilot studies demonstration and evaluation
    - As its main outcome, the validation test cases provide WP10 pilots with a functional instance of the integrated system to support and validate, in turn, the CoRoSect use cases in the selected insect farms.

1.3. Structure of the document

The CoRoSect ecosystem is built by two main divided blocks that correspond to i) the I4.0 IT systems, composed by the Shop Floor devices; and ii) the OT systems, represented by the CoRosect's MES. With this approach, the Shop Floor devices will cover all the mechatronics and IoT devices to be deployed in the farms' scenarios whilst the MES includes the IMS, the SFM and the DSS components. This structure is followed to organise and resume the set of test cases within this document:

- **Section 1** - Introduction (this section). Introduces the objectives, scope, and outcomes for task 9.3 summarised for this first task's deliverable, plus showing and justifying the documents' structure to resume these contents.
- **Section 2** - CoRoSect integrated System scenario. Describes the instances of the software components, linked, in case, to their hardware devices (corresponding robots, IoT systems, cameras, etc.) used to run the validation test cases and the corresponding interfaces checked.
- **Section 3** - Integration tests. Covers the CoRoSect Ecosystem test bed for the Shop floor components including corresponding properties (data sharing) and operations (commands' flows) for:
    - Stacking/De-staking Robot (D-Robot)
    - Manipulation Robot (M-Robot)
    - Intelligent Crates (I-Crates)
    - Automated Guided Vehicle (AGV)
    - HoloLens System
    - Route Manager
    - Objects detector
- **Section 4** - Functional tests. Deals with the functional validation of data distribution and data presentation addressed to the CoRoSect MES through its IMS.
- **Section 5** - Simple orchestration tests. Validates the Shop Floor Manager (assisted by the Decision Support System) capability to command the CoRoSect's Shop Floor.
- **Section 6** – Conclusions. Finally, presents the main outcomes and conclusions of the validation test cases executed.

2. CoRoSect integrated system testbed

2.1. CoRoSect's distributed instance

All test cases were run in a distributed way: each shop floor component, composed by the physical device plus its corresponding server (either OPC-UA or MQTT Client) and/or its customised simulator is executed within the provider's premises. Using TCP (supporting OPC and MQTT), these servers and simulators connect with the cloud IMS instance, which acts as the data hub and commands' driver. In turn, the Shop Floor Manager and the DSS work in this way, using the IMS as the gateway to access the shop floor components. The schema for this distributed testbed is shown in Figure 2.

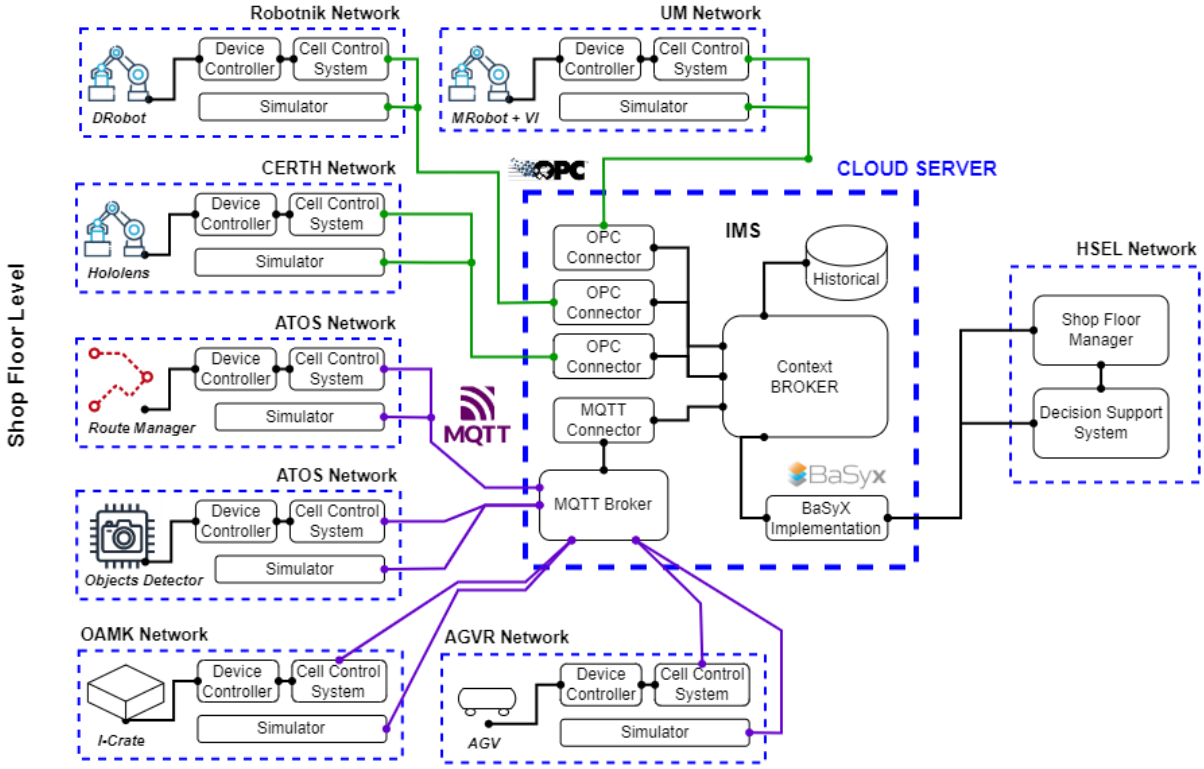

Figure 2. Distributed testbed for CoRoSect's System validation

The implementation of this test bed corresponds to the **Integrated CoRoSect System version 1** instance detailed in D9.2 (Figure 3) and is according to the CoRoSect System Architecture presented in D2.4.

The purpose of the testbed is to validate the communications between the implemented servers using the interfaces and protocols defined following RAMI4.0 requirements. This is to enable the work done during the first stage of the project, designing the system architecture and the integration interfaces, for their use in the CoRoSect pilots and, in turn, the validation of the functional requirements defined by WP2. According to the system architecture and its implementation (Figure 3), we have divided the testbed (Figure 2) in three main blocks:

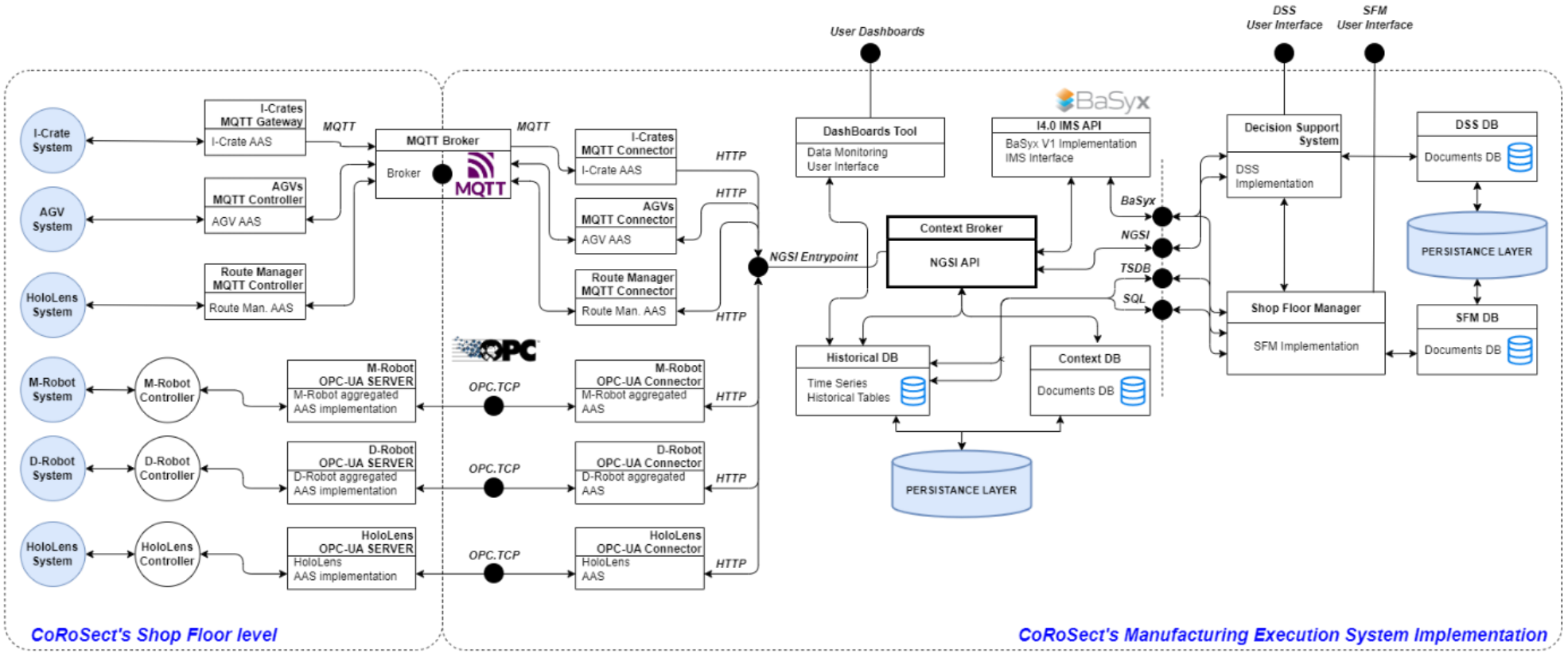

Figure 3. Integrated CoRoSect System (V1) implementation (From D9.2)

- **The Shop Floor**, covering all the CoRoSect's shop floor components. This includes the servers implemented to connect the CoRoSect MES with the corresponding devices. Each device and cell controller will run in the provider's network and will connect to the cloud IMS instance using TCP. Here we can distinguish two sets of shop floor devices:
    - Those using **OPC-UA on top of TCP**. They use a specific developed OPC-US server and communicates with the core system through an OPC connector developed within CoRoSect's scope. These are:
        - The Manipulation Robot (**M-Robot**) also embedding the Visualization Module (**VI**). This is provided by UM (the M-Robot) plus CERTH (VI) and runs in the UM premises.
        - The Stacking/Destacking Robot (**D-Robot**) provided by Robotnik and running within its network infrastructure.
        - The **HoloLens** device supported by CERTH and connected from their premises.
    - Those relying on **MQTT protocol driven by TCP**. They implement an MQTT client to publish/subscribe (update data/get commands) in the MQTT broker provided by the IMS. They are:
        - The **I-Crates** and their sensors infrastructures, supported by OAMK.
        - The **Routes' Manager** and the **Objects' detector** developed and tested by ATOS (merged in this V1 version in the Routes' Manager MQTT module).
        - The **AGV** provided by AGVR.
- The **Shop Floor Manager** (SFM) and the **Decision Support System** (DSS) designed and implemented by HSEL. Their structure and functionalities are introduced in D4.1 and D4.2, and their final description will be shown as part of the MES system in D4.3. Shop Floor Manager accesses the IMS using the CoRoSect's customised implementation of the Eclipse BaSyx[1] interface. This is supported by an HTTP connection.
- The CoRoSect's **Information Management System** (IMS) infrastructure is deployed in a **cloud server** managed by ATOS. It is built on top of a Kubernetes cluster and all CoRoSect partners have access to it. This way, the IMS connects with all shop floor components through the corresponding interfaces to collect and distribute shop floor data. It also enables the command and control of the shop floor devices by the SFM/DSS using I4.0 protocols and standards. IMS relays on the RAMI4.0 interfaces and Asset Administration Shells defined in Task 9.1. Further details of this component will be provided in D4.3.

2.2. Subsystem interfaces (Digital Twins)

The integration test cases executed and described within this text also target the validation of each interface implementation carried out by each shop floor provider and so, the corresponding Digital Twin (DT) created in the IMS. Each of the DTs maps all the relevant parameters and commands for its physical device, following the *RAMI 4.0 Layer 4 compatible technology* by using the Asset Administration Shell (AAS) and abstracting its hardware implementation from the eyes of the SFM. The CoRoSect's DTs thus constitute an implementation of the I4.0 Asset Administration Shell (AAS) defined by Task 9.1 (described in D9.1). The Digital Twin full structure and the way it is embedded within the IMS are the contents of the D4.3. To better illustrate the tests cases, we advance here the main concepts of their structure. Each DT can be represented as an entities' tree composed by:

[1] https://eclipse.dev/basyx/

- The root level is the **Asset Administration Shell** (AAS) entity, which details the structure and hierarchy of the required data and information, with the reference links, of the set of sub-models that represents all the capabilities and data offered by the device/system.
- A descriptive branch includes the **Asset** entity, which mainly contains physical and descriptive information that refers the device/system in the real world.
- Of the AAS hangs the different **Sub-models** entities. As mentioned in D9.1, "*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"*. These Sub-models articulate the actual digitalization of the device/system as they support all the data and commands transactions. Each sub-model corresponds to a specific characteristic (or set of related ones), a functionality and/or a command supported/offered by the system, known internally as "Element". Within CoRoSect, each sub-model can group different "Elements" as Sub-model Element entities.
- The Sub-model Elements represent the characteristics and attributes of each sub-model, which finally map the capabilities of the digitalised device. We manage three different Sub-model Element types:
    - Properties (**SubmodelElementProperty**): represents an attribute of the system/device (e.g., location, temperature, status, etc.), including its value and the metadata needed to read and interpret it. These properties are usually updated by the device system controller in the IMS, periodically or as a response to a related command.
    - Operations or Commands (**SubmodelElementOperation**): maps a concrete operation supported by the device that can be triggered by the SFM. It includes both, the input variables (or values) required to execute the command; and the output (if the output is offered also through this element) for the call. Usually, the SFM will update these *SubmodelElementOperation* entities by writing their input variable/s. In this way, the IMS transmits to the device' system controller both, the command, and the parameters.
    - Ranges (**SubmodelElementRange**): It represents attributes as the SubmodelElementProperty element with the possibility of defining two possible values for a given attribute/s. In this case, a **SubmodelElementRange** can be used to define a minimum and maximum value where the represented attribute can fluctuate.

Figure 4 shows an example of this Digital Twin structure, specifically the one created to map the Manipulation Robot (plus Visual Inspection module). The rest of DTs follow a similar structure and will be presented in D4.3, as they also represent the Information Model core of the CoRoSect MES. The IMS used within the testbed for the testcases defines Digital Twins for:

- Manipulation Robot + Visual Inspection module (**MRobotVI**)
- Staking/Destacking Robot (**DRobot**)
- CoRoSect's Microsoft HoloLens system (**Hololens**)
- Route Management system (**RManager**)
- Objects' Detector system (**ObjectDetector**)
- Pilot's Automated Guided Vehicle (**AGV1**)
- One of the Intelligent Crates developed for the pilots (**ICrate**)

These DTs support the interfaces and the data and commands flows used in the test cases described in Section 3.

As shown in Figure 2, the shop floor providers have also designed simulators for their corresponding systems. These simulators have been developed to support the device AAS interface, exposing the set of properties and operations which will be finally implemented by the actual physical system. The purpose of these simulators is to mock-up the RAMI4.0 compliant responses to SFM commands and data updates through the IMS. This is useful for the refinement and testing of the IMS connectors and the implementation of the corresponding systems' DTs.

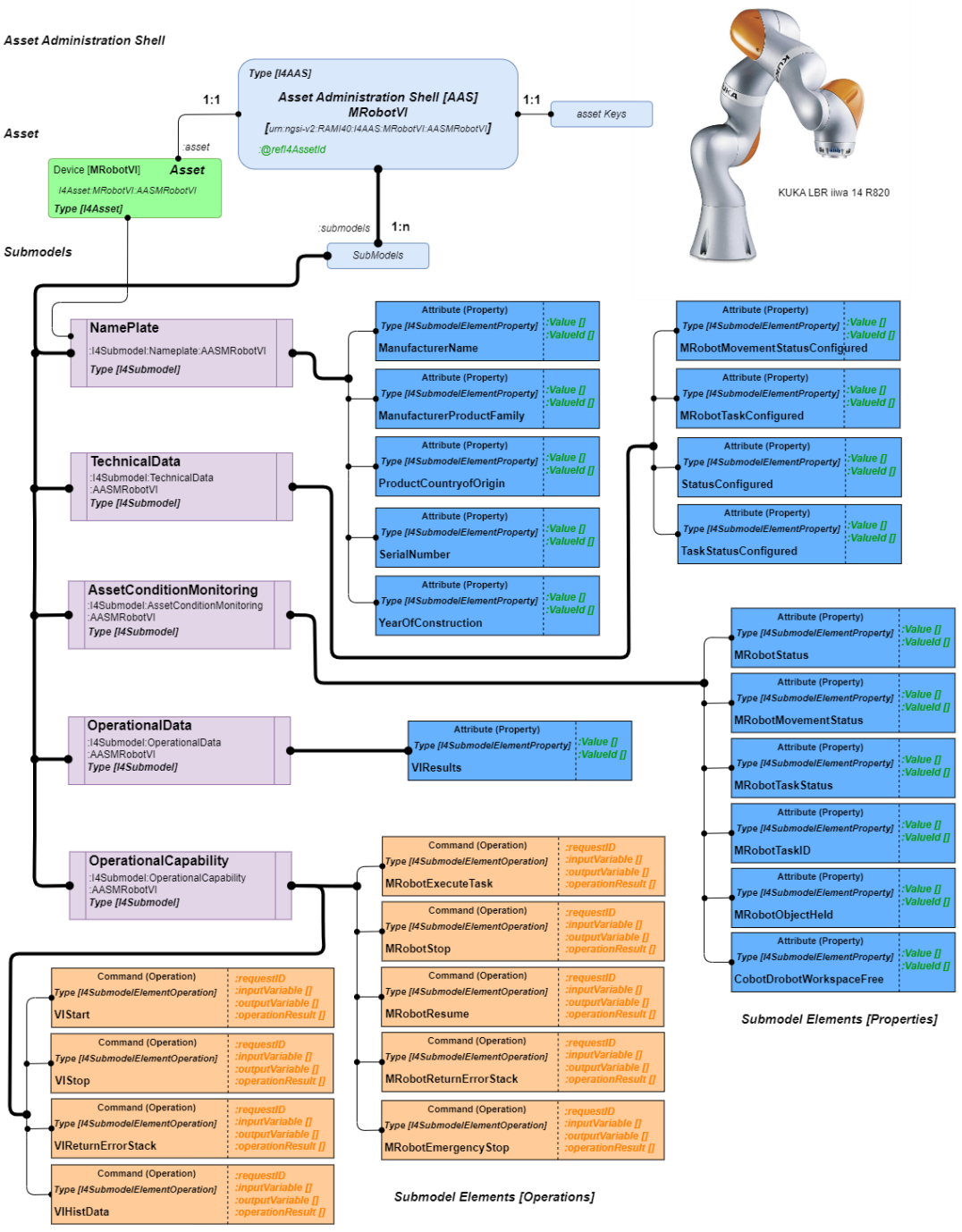

Figure 4. Example of the Digital Twin structure defined within CoRoSect. Represents the MRobotVI DT.

For each shop floor component listed in D9.1 and D9.2, the corresponding BaSyx API calls will be executed to ensure that each of the AAS sub-models and sub-models' elements defined on its corresponding AAS are present on the Digital Twin created in the IMS and accessible, as well as the reported values.

3. Integration tests cases

3.1. Overall validation scenario

Task 9.3 test cases are intended to verify the **proper integration of each CoRoSect Shop Floor component** with the project's centralised **Manufacturing and Execution System** (MES). This is done through the Information Management System (IMS) and according to the components' corresponding Asset Administration Shell (AAS) (D9.1 Interfaces). The targets are:

1. to check the **data availability** and **accessibility** (Data Query/Retrieve) reported by each component's interface, ensuring the data update process triggered by each component;
2. to ensure the proper **calling** of each of the listed **commands** (Commands Call/Retrieve) supported by each component and their proper responses through the IMS; and
3. to refine the adoption and implementation (data formats, adherence to data models, use of protocols and commands executions) of the mentioned interfaces to ensure I4.0 compatibility of the different DT supported by the RAMI 4.0.

The evaluation point is set on the RAMI4.0 compliant BaSyx interface developed for the IMS which works as the communication bridge between the IMS and the pair SFM and DSS. It acts as the communication hub within the CoRoSect's MES (Figure 5). This REST interface exposes the HTTP endpoints (and methods) to:

i) read AAS entities and structures stored in the IMS (access all registered Digital Twins)
ii) read (query/retrieve) sub-models and sub-model elements (properties) for all registered AAS/DTs in the IMS, retrieving all corresponding values and linked metadata (execute synchronous data query/retrieve)
iii) get subscribed to all/selected elements (properties and operations) to automatically receive any update on them (implements asynchronous data sharing through publish/subscribe)
iv) invoke (execute) any sub-model element property linked to a registered DT, triggering the execution of the corresponding command in the shop floor device.

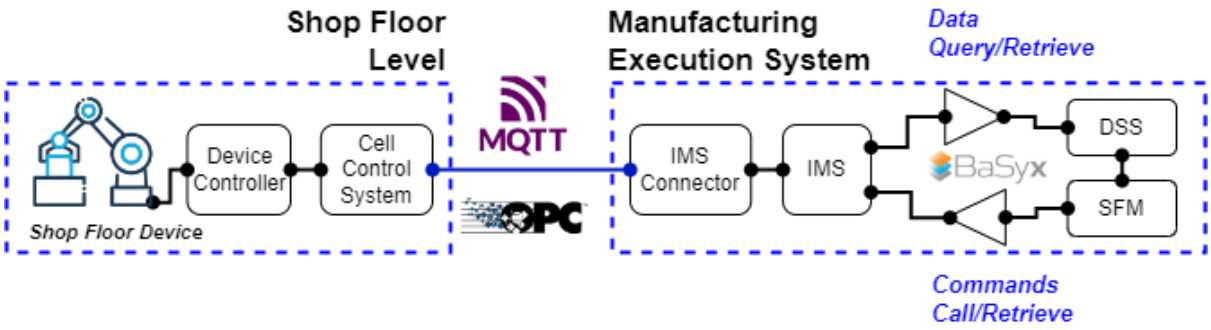

Figure 5. General approach to the integration test cases, focused on the CoRoSect's BaSyx interface.

With this approach and by controlling when and what each shop floor system produces and updates attributes on its local cell controller:

- By **synchronously query/retrieve** updated data to the IMS through its BaSyx interface we validate both: a) the interface implementation and the corresponding mapped Digital Twin; and b) the proper communication between the component's Cell Control System and the corresponding connector (either MQTT or OPC-UA), as well as their software implementations.

- By receiving asynchronous notifications (on data updates) we are also validating a) the communication between the local shop floor device and the IMS; and b) the implementation of the **asynchronous data sharing paths**.

By controlling the physical execution of the commands triggered by the SFM we:

- Validate the **commands' flows** between the SFM and the corresponding shop floor device, checking the full path: i) BaSyx's Invoke endpoint; ii) operation mapping (structure, input/output parameters, etc.) on the DT; iii) IMS connection (specific developed connector) with the cell controller; and iv) the proper translation of the input values and instruction received to the proprietary system protocols.
- Also, the responses' updates from the system cell controller reinforce the validation of the **communication protocols implementation**, the **deployed DTs** and the **data sharing paths** (synchronous and asynchronous).

The integration test cases for CoRoSect system have three steps in common that are customised later to each property and operation to be checked. These are: i) the data uploading and updating from the shop floor devices towards the IMS, to validate the synchronous data query/retrieve processes; ii) the notifications to the SFM when new subscribed data arrives, to check the asynchronous data sharing; and iii) the commands triggering and responses retrieval, to complete the DT validation and the functionalities of the MES. These steps are repeated for each system validation test. Their common schemas are shown here.

3.1.1. Data Uploading/Updating from Shop Floor Level. Synchronous data query/retrieve.

Deliverables from WP4 (D4.1 and D4.3) and interfaces (D9.1) and CoRoSect's system implementation (D9.2) describe in detail the processes designed to integrate the data uploading and updating from the Shop Floor components.

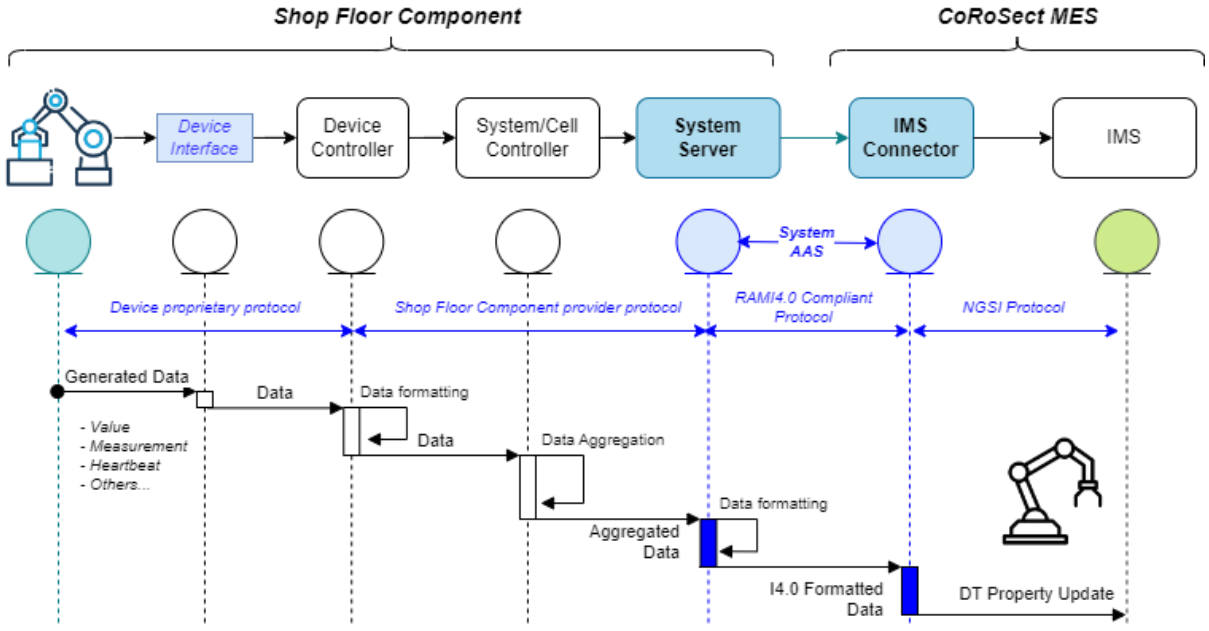

Figure 6. Common data upload/update from the Shop floor device to the IMS, reflected in the corresponding Digital Twin property.

Figure 6, expanded from D9.2, describes the process followed by each shop floor system to update its CoRoSect Digital Twin with data generated (and captured) by the physical device. This data is registered and stored in the corresponding sub-model element property, to be served when requested by the SFM (or any other CoRoSect system), or automatically notified to registered systems.

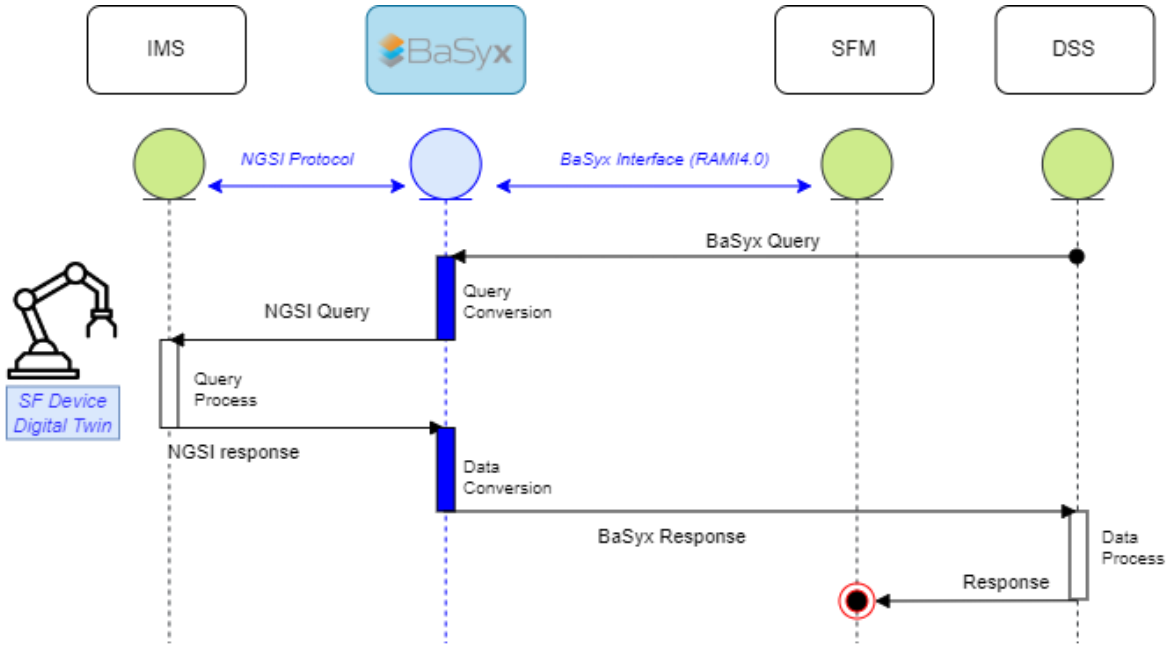

Figure 7. Common data query to the IMS (from SFM to IMS).

Figure 7 depicts the synchronous data query triggered from the CoRoSect's MES (SFM+DSS). This process is replicated by each test case in sections 3.2 and beyond, intended for validating: i) the proper data update and data accessibility; ii) the proper implementation of the BaSyx corresponding method; and iii) the correct implementation and use of the device's digital twin.

The DSS/SFM pair a) launches the request by calling the corresponding BaSyx HTTP method (RAMI4.0 compliant) to query an AAS, a given Sub-model or a concrete Sub-model Element (mainly properties); b) the IMS BaSyx module receives the request and converts it into an NGSI proper query, c) to be managed within the CoRoSect's IMS; d) the IMS retrieves the requested information and sends it back to the BaSyx module, e) which adapts the data and its format to the BaSyx's response; f) This is then sent to the DSS, which processes it and g) caters it to the SFM.

Retrieve Sub-model		
GET	[*BaSyx Server*]/aas/[AAS_ID]/submodels/[Submodel_ID]	
Headers	Fiware-Service	Define isolated environments where all defined DTs and datasets are related. Can refer to e.g a testing environment or map all farm's deployed systems
	Fiware-ServicePath	
Payload	No payload	
Return values		
code	404	AAS and/or Submodel not found
code	50X	Server error
code	200/201	Success:
		Retrieves JSON object including: Submodel description + list of linked submodel elements (properties, ranges and/or operations).

Retrieve Submodel Elements' VALUES (for Submodel Elements' PROPERTIES)		
GET	[*BaSyx Server*]/aas/[AAS_ID]/submodels/[Submodel_ID]/ submodel/submodelElements/[SubmodelElement_ID]/**value**	
Headers	Fiware-Service	Define isolated environments where all defined DTs and datasets are related. Can refer to e.g., a testing environment or map all farm's deployed systems
	Fiware-ServicePath	
Payload	No payload	
Return values		
code	404	AAS and/or Submodel and/or Submodel Element Property not found
code	50X	Server error
code	200/201	Success:
		Retrieves JSON object including ***value***, *valueType* and *valueId* for the referred property.

Important Note: The corresponding IDs refer to the unique identifier defined in the corresponding Asset Administration Shell (AAS) as "IdShort".

3.1.2. Asynchronous Data retrieval

The asynchronous data retrieval tests guarantee that the latest data reported from the Shop Floor level is notified to the Shop Floor Manager/Decision Support System immediately. To achieve this, the IMS implements a Publish-Subscribe mechanism, supported by an NGSI Context Broker. Using this middleware, the IMS will automatically send a POST notification to a registered HTTP endpoint including the new information of interest every time a new update on this information is received, without the interested system must request for it.

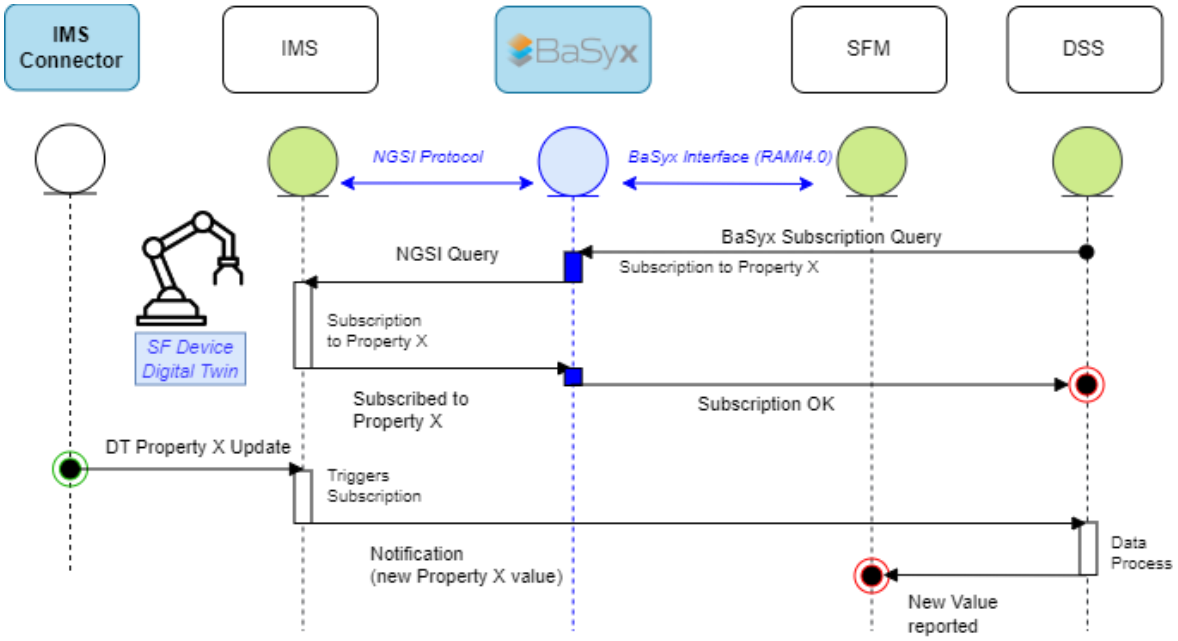

Figure 8. CoRoSect's system subscription and notification process description.

Figure 8 depicts the publish/subscribe process within CoRoSect's system to be asynchronously notified when "Property X" changes. The Shop Floor Manager a) request a subscription to Property X (of a selected DT) to the BaSyx module, subscribing the DSS (to receive notifications) as its HTTP endpoint; b) the BaSyx module converts and redirects the subscription request (and parameters) to the IMS core, using NGSI API; c) the IMS broker registers the subscription and the referred endpoint and returns and "201-OK" (success) HTTP response, which is d) redirected to the SFM. When e) "Property X" is updated in the IMS, the IMS f) checks its list of active subscriptions and sends the proper HTTP notification,

with the new values for “Property X” to the registered DSS endpoint; g) the DSS receives this new information and acts according to its program.

Following tables list the BaSyx methods to work with the CoRoSect’s Asynchronous data retrieval: i) create and register a subscription; and ii) Notification HTTP POST message.

Create and register a generic Subscription (using NGSI payload)		
POST	[*BaSyx Server*]/subscription	
Headers	Fiware-Service Fiware-ServicePath	Define isolated environments where all defined DTs and datasets are related. Can refer to e.g a testing environment or map all farm’s deployed systems
Payload	JSON with Subscription NGSI payload	{ "description":"Subs to Property X", //describes the subs’ target "subject":{ //defines ’what’ you are subscribing to "entities":[{ "idPattern":"Property X:DT Y", //Property X (from DT Y) "type":"I4SubmodelElementProperty" //NGSI valuetype Property }], "condition":{ "attrs":["value”] //when ”value” changes } }, "notification":{ //send an http notification (POST) to "http":{ "url":"http://dsshost:dssport/dss/notifications " }, "attrs": ["temperature"] //including ”value” attribute } }
Return values		
code	400	Bad Request. Returns a description with the detected error
code	50X	Server error
code	200/201	Success (Subscription properly executed)
		Header: subsID. Returns the subscription ID to later operate (modify/delete) with it.

NOTIFICATION		
POST	[*http notification url*]	
Headers	Fiware-Service Fiware-ServicePath	Define isolated environments where all defined DTs and datasets are related. Can refer to e.g a testing environment or map all farm’s deployed systems
Payload	NGSI JSON Notification payload ref[2]	{ "subscriptionId": "12345", "data": [{ "id": "PropertyX:DT_Y", "type": "I4SubmodelElementProperty", "Value": { "value": 1, "type": "Number", "metadata": {} } }] }
Return values (depends on the service implemented by the receiving endpoint, but MUST include)		

[2] https://github.com/telefonicaid/fiware-orion/blob/master/doc/manuals/orion-api.md#notification-messages

code	50X	Server error
code	200/201	Success (notification received)

3.1.3. Commands' calling and responses' retrieval

For each shop floor component listed in D9.1 and D9.2, the corresponding BaSyx HTTP REST API call **INVOKE operation** is executed to call (and trigger) each of their sub models' operations (commands) defined on its corresponding AAS. The expected responses, mapped as changes on the corresponding properties, will be also checked.

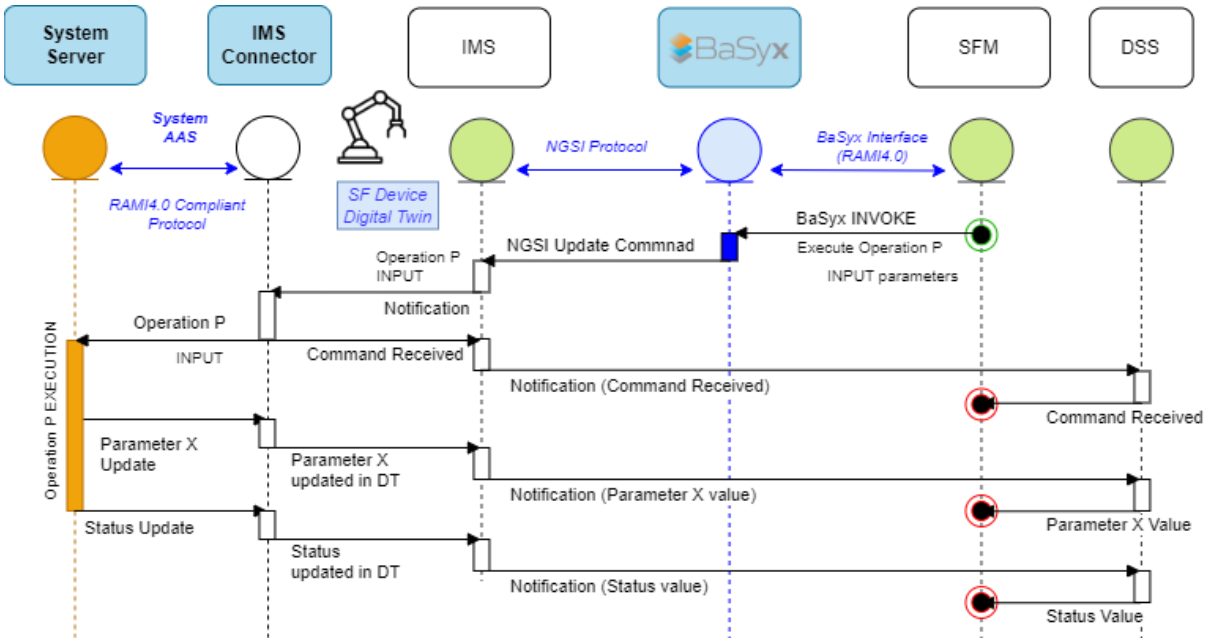

Figure 9. BaSyx INVOKE HTTP resource to request the execution of Command/Operation P.

Figure 9 represents the flow to request the execution of the Operation P, belonging to an already defined Digital Twin within the IMS. The SFM a) calls the HTTP POST INVOKE resource offered by the CoRoSect's BaSyx module including the "*inputVariables*" payload; the BaSyx module b) converts the request into the corresponding NGSI Update command, so the c) IMS can write in the selected Submodel Element Operation of the DT; with this operation, d) the IMS corresponding connector is notified and the input parameters are e) sent to the device system server. Once received, the server f) communicates with the shop floor through the driver to execute the requested command with the given input variables. During the execution process, g) the driver, and so the system, can update one or several device's parameters, which will be properly updated in the DT and communicated to the SFM following the Asynchronous data sharing previously described. Finally, when the command has finished, h) the device status is updated and reported in the Asynchronous way.

The **SubmodelElement_ID** parameter identifies the command (Submodel Element Operation) to be executed.

INVOKE Operation (Command call)		
POST	[*BaSyx Server*]/aas/[AAS_ID]/submodels/[Submodel_ID]/submodel/ submodelElements/[SubmodelElement_ID]/**invoke**	
Headers	Fiware-Service	Define isolated environments where all defined DTs and datasets are related. Can refer to e.g a testing environment or map all farm's deployed systems
	Fiware-ServicePath	

Payload	JSON with required inputs to execute the command	{
		"requestId":"0A1B2C3D4", //to track the command
		"timeout": 100,
		"inputArguments": [], //required inputs to execute the command
		"inoutputArguments": []
		}

Return values		
code	404	AAS and/or Submodel and/or Submodel Element not found
code	50X	Server error
code	200/201	Success:
		"Operation invoked successfully"

If the INVOKE operation has been successful, a response (OutputVariable) and/or different status properties may be updated on the component's Digital Twin, depending on the implementation (and execution) of the called command. To read these updates, the following calls can be used.

Retrieve Submodel Element Operation OUTPUT (for Submodel Elements' OPERATION)		
GET	[*BaSyx Server*]/aas/[AAS_ID]/submodels/[Submodel_ID]/ submodel/submodelElements/[SubmodelElement_ID]/**output**	
Headers	Fiware-Service Fiware-ServicePath	Define isolated environments where all defined DTs and datasets are related. Can refer to e.g a testing environment or map all farm's deployed systems
Payload	No payload	
Return values		
code	404	AAS and/or Submodel and/or Submodel Element Property not found
code	50X	Server error
code	200/201	Success:
		Retrieves JSON object including: ***outputVariable*** for the referred operation.

Retrieve Submodel Elements' VALUES (for Submodel Elements' PROPERTIES that may changed with the command request)		
GET	[*BaSyx Server*]/aas/[AAS_ID]/submodels/[Submodel_ID]/ submodel/submodelElements/[SubmodelElement_ID]/**value**	
Headers	Fiware-Service Fiware-ServicePath	Define isolated environments where all defined DTs and datasets are related. Can refer to e.g a testing environment or map all farm's deployed systems
Payload	No payload	
Return values		
code	404	AAS and/or Submodel and/or Submodel Element Property not found
code	50X	Server error
code	200/201	Success:
		Retrieves JSON object including ***value***, *valueType* and *valueId* for the referred property.

This BaSyx API endpoint and its REST resources can also be offered to external applications/information systems, as a RAMI 4.0 interface for further integrations.

Next subsections use the mentioned methods in the current section to query each Digital Twin listed properties and invoke their supported operations according to their corresponding AAS defined in D9.1. This way we validate the CoRoSect Integrated system for the next step: CoRoSect Pilots.

Important Note: we've focused these test cases on the CoRoSect's pilots and the use cases to be evaluated. For this purpose, not all properties and operations listed on the mentioned AAS are mandatory to be implemented, so we enable each device/system for the CoRoSect pilots when all the required set of properties and operations are properly checked. Any missing (not mandatory) sub-model element (either property or operation) will be implemented during the progress of the different pilots.

Important Note: The corresponding IDs refer to the unique identifier defined in the corresponding Asset Administration Shell (AAS) as "IdShort".

For properties

Tables in sections 3.2 and beyond check the availability of each of the implemented values, supported and provided by each of the devices' servers (either on top of OPC or MQTT) according to their AAS interfaces by calling the *GET Submodel Element API REST* resource provided by the BaSyx interface.

HTTP API REST (BaSyx) Retrieve Submodel Elements' VALUES		
GET	[*BaSyx Server*]/aas/DRobot/submodels/[Submodel_Name]/ submodel/submodelElements/[Submodel_Element]/**value**	
Headers	Fiware-Service	Define isolated environments where all defined DTs and datasets are related. Can refer to e.g a testing environment or map all farm's deployed systems
	Fiware-ServicePath	
Payload	No payload	
Return values		
code	404	AAS and/or Submodel and/or Submodel Element/ID Property not found
code	50X	Server error
code	200/201	Success:
		Retrieves JSON object including ***value***, *valueType* and *valueId* for the referred property.

Tables present a column to reflect if each listed property is supported by its simulator (in case the device provides a simulator) and another column to verify if the property is supported by (and has been properly read from) the real device.

3.2. Stacking/De-staking Robot (D-Robot) validation

Staking/De-staking Robot (D-Robot) (Figure 10) is consisted of a Kuka KR70 robotic arm, a Schunk PSH52 gripper and a camera (for visual and reliable control purposes). This is further detailed in D9.2.

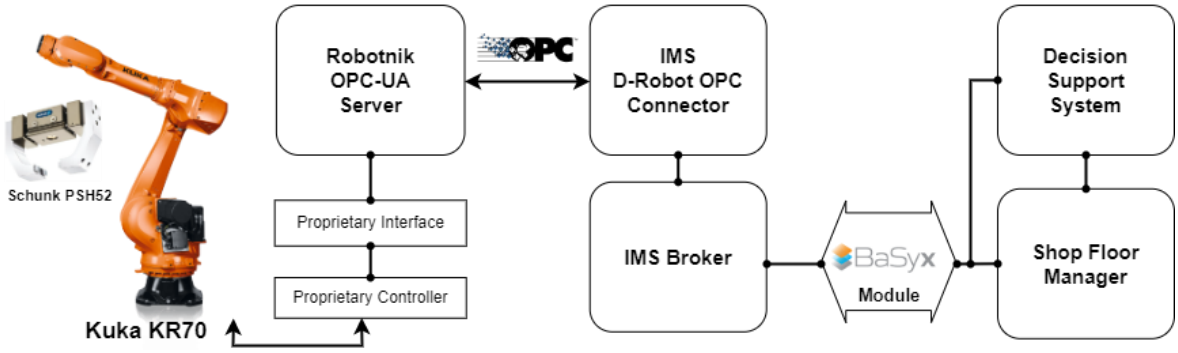

Figure 10. D-Robot integration scenario

D-Robot works with OPC-UA and implements its own OPC-UA Server to support its full D9.1 defined AAS. Robotnik provides (and supports) this OPC-UA server and also a D-Robot OPC-UA simulator, used to test the proper OPC-UA connectivity.

3.2.1. Properties' integration tests

These tests (Table 1) use the Synchronous Query/Retrieve process (Figure 7) to validate the D-Robot data uploading process and data availability through the IMS. Retrieving times are indicative (as they vary from one request to another) and represents the average time after having performed several requests. These times refer to the CoRoSect's cloud instance.

Table 1. Properties' availability tests for D-Robot

AAS:		SUBMODEL		Simulator	Real Device/System	
DRobot		Name	Element	Read	Read	Retrieving time
DATA (Properties)	Values	AssetConditionMonitoring	RobotMovementStatus	Checked	Checked	396 ms
			Status	Checked	Checked	396 ms
			GripperObjectHeld	Not Supported	Not Implemented	
			TaskStatus	Checked	Checked	396 ms
			TaskID	Checked	Checked	396 ms
			DRobotCobotWorkspaceFree	Not Supported	Checked	396 ms
			DRobotAGVSharedWorkspaceFree	Not Supported	Not Implemented	
		TechnicalData	GeneralTasksConfigured	Checked	Checked	396 ms
			SpecificTasksConfigured	Checked	Checked	280 ms
			StatusConfigured	Checked	Checked	280 ms
			RobotMovementStatusConfigured	Checked	Checked	280 ms
			TaskStatusConfigured	Checked	Under Review	
		Nameplate	ManufacturerName	Checked	Checked	266 ms
			ManufacturerProductDesignation	Checked	Checked	266 ms
			Country code	Checked	Checked	266 ms
			Street	Checked	Checked	266 ms
			Zip	Checked	Checked	266 ms
			CityTown	Checked	Checked	266 ms
			StateCounty	Checked	Checked	266 ms
			ManufacturerProductFamily	Checked	Checked	266 ms
			YearOfConstruction	Checked	Checked	266 ms
			SerialNumber	Checked	Checked	266 ms
			ClassificationSystem	Checked	Checked	266 ms
			DateOfManufacture	Checked	Checked	266 ms
			ProductCountryOfOrigin	Checked	Checked	266 ms
			QrCode	Checked	Checked	266 ms
			ProductIdentifier	Checked	Checked	266 ms

3.2.2. Operations' triggering tests

These tests (Table 2 to Table 7) use the INVOKE HTTP process (Figure 9) to validate the D-Robot connectivity with the Shop Floor Manager. This validates the availability of the D-Robot to be commanded by the SFM and fully integrated with the CoRoSect MES. Retrieving times are indicative (as they vary from one request to another) and represents the average time after having performed several requests. These times refer to the CoRoSect's cloud instance.

Table 2. List of supported commands from D-Robot AAS

AAS: DRobot	SUBMODEL Name	Element	Simulator	Real Device/System
COMMANDS (Operations)	OperationalCapabilities	ExecuteGeneralTask	Supported	Supported
		DeStackCrate	Supported	Supported
		StackCrate	Supported	Supported
		StopOperation	Not Supported	Not Implemented
		ResumeOperation	Not Supported	Not Implemented
		ReturnErrorStack	Not Supported	Not Implemented
		EmergencyStop	Supported	Supported
		InitializeCell	Supported	Supported

Table 3. ExecuteGeneralTask command test for D-Robot

COMMANDS (Operations)	Operational Capability	**Submodel Element**		**ExecutegeneralTask**		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**DRobot**/submodels/**OperationalCapability**/submodel/submodelElements/**ExecuteGeneralTask**/invoke		
			InputVariable	"inputVariable": [{"value": 1,"idShort": "TaskID"}]		
			Output		**Simulator**	**Real Device/System**
		Results	"operationResult": { "success": true, "isException": false }, "outputVariable": [{ "value": true }]		Checked	Checked
			Properties updated	AssetConditionMonitoring.TaskID	Checked	ID of the Op (100)
				AssetConditionMonitoring.TaskStatus	Checked	Status: (1 [Exec])- 2 [Success] – 3 [Fail]
				AssetConditionMonitoring.RobotMovementStatus	Checked	0 (moving) 1(stopped) 2(emergency stop)
				AssetConditionMonitoring.DRobotCobotWorkspaceFree	Checked	0/1 (in/out manipulation zone)
		Errors	Description	Command's failure (Robot failure)	Checked	Checked
			Output Result	"operationResult": { "success": false, "isException": true }		

Table 4. DeStackCrate command test for D-Robot

COMMANDS (Operations)	Operational Capability	**Submodel Element**		**DeStackCrate**		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**DRobot**/submodels/**OperationalCapability**/submodel/submodelElements/**DeStackCrate**/invoke		
			InputVariable	"inputVariable": [{ "value": {"x": 0, "y": 0, "z": 0}, "idShort": "CrateLocation" }]		
			Output		**Simulator**	**Real Device/System**
		Results	"operationResult": { "success": true, "isException": false }, "outputVariable": [{ "value": true }]		Checked	Checked
			Properties updated	AssetConditionMonitoring. TaskID	Checked	ID of the Op (102)
				AssetConditionMonitoring. TaskStatus	Checked	Status: (1 [Exec])- 2 [Success] – 3 [Fail]
				AssetConditionMonitoring. RobotMovementStatus	Checked	0 (moving) 1(stopped) 2(emergency stop)
				AssetConditionMonitoring. DRobotCobotWorkspaceFree	Checked	0/1 (in/out manipulation zone)
		Errors	Description	Command's failure (Robot failure)	Checked	Checked
			Output Result	"operationResult": { "success": false, "isException": true }		

Table 5. StackCrate command test for D-Robot

COMMANDS (Operations)	Operational Capability	**Submodel Element**		**StackCrate**		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**DRobot**/submodels/**OperationalCapability**/submodel/submodelElements/**StackCrate**/invoke		
			InputVariable	"inputVariable": [{ "value": {"x": 0, "y": 0, "z": 0}, "idShort": "CrateLocation" }]		
			Output		**Simulator**	**Real Device/System**
		Results	"operationResult": { "success": true, "isException": false }, "outputVariable": [{ "value": true }]		Checked	Checked
			Properties updated	AssetConditionMonitoring. TaskID	Checked	ID of the Op (101)
				AssetConditionMonitoring. TaskStatus	Checked	Status: (1 [Exec])- 2 [Success] – 3 [Fail]
				AssetConditionMonitoring. RobotMovementStatus	Checked	0 (moving) 1(stopped) 2(emergency stop)
				AssetConditionMonitoring. DRobotCobotWorkspaceFree	Checked	0/1 (in/out manipulation zone)
		Err	Description	Command's failure (Robot failure)	Checked	Checked

			Output Result	"operationResult": {
				"success": false,
				"isException": true
				}		

Table 6. EmergencyStop command test for D-Robot

COMMANDS (Operations)	Operational Capability	**Submodel Element**		**EmergencyStop**		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**DRobot**/submodels/**OperationalCapability**/submodel/submodelElements/**EmergencyStop**/invoke		
			InputVariable	"inputVariable": [{ "value": "None", "idShort": "None", "descriptions": [{"language": "en", "text": "No input required for Emergency Stop"}] }]		
			Output		**Simulator**	**Real Device/System**

		Results		"operationResult": {
				"success": true,
				"isException": false
				},
				"outputVariable": [{
				"value": true
				}]
					Checked	**Checked**

			Properties updated	AssetConditionMonitoring. TaskID	**Checked**	**ID of the Op (105)**
				AssetConditionMonitoring. TaskStatus	**Checked**	**Status: (1 [Exec])- 2 [Success] – 3 [Fail]**
				AssetConditionMonitoring. RobotMovementStatus	**Checked**	**2 (emergency stop)**
				AssetConditionMonitoring. DRobotCobotWorkspaceFree	**Checked**	**0/1 (in/out manipulation zone)**
		Errors	Description	Command's failure (Robot failure)	**Checked**	**Checked**

			Output Result	"operationResult": {
				"success": false,
				"isException": true
				}		

Table 7. InitializeCell command test for D-Robot

COMMANDS (Operations)	Operational Capability	**Submodel Element**		**InitializeCell**		
		INVOKE operation	REST call (POST)	[BaSyx_Server]/aas/**DRobot**/submodels/**OperationalCapability**/submodel/submodelElements/**InitializeCell**/invoke		
			InputVariable	"inputVariable": [{ "value": {"x": 2, "y": 2, "z": 1}, "idShort": "GridDescription" }]		
			Output		**Simulator**	**Real Device/System**

		Results		"operationResult": {
				"success": true,
				"isException": false
				},
				"outputVariable": [{
				"value": true
					Checked	**Checked**

			}]			
		Properties updated	AssetConditionMonitoring. TaskID		Checked	ID of the Op (106)
			AssetConditionMonitoring. TaskStatus		Checked	Status: (1 [Exec])- 2 [Success] – 3 [Fail]
		Errors	Description	Command's failure (Robot failure)	Checked	Checked
			Output Result	"operationResult": { "success": false, "isException": true }		

3.3. Manipulation Robot + Visual Inspection module (M-RobotVI)

The Manipulation Robot (M-Robot), provided by UM, includes a Visual Inspection Module supported by CERTH and so conform the Manipulation and Visual Inspection Robot (M-RobotVI) (Figure 11). It is composed by a Kuka LBR iiwa14 R820 robotic arm with the corresponding gripper and tools and a set of industrial cameras connected through USB port to the robot controller. This is further detailed in D9.2.

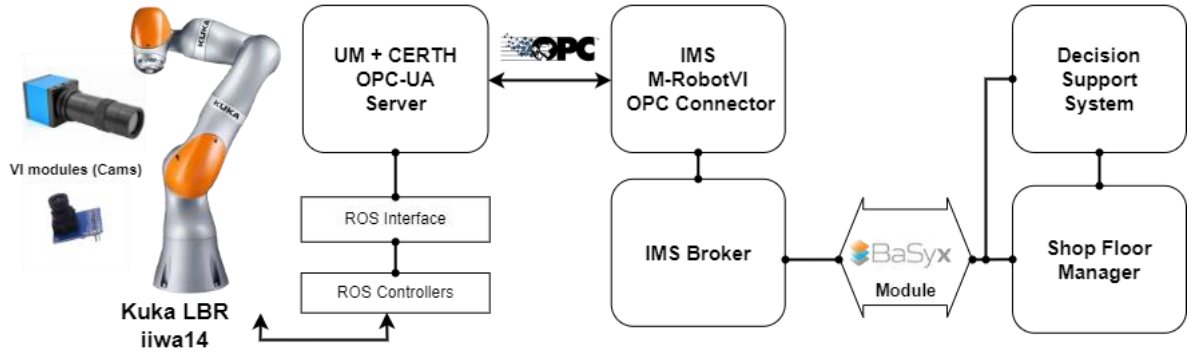

Figure 11. M-RobotVI integration scenario

M-RobotVI set works using an integrated OPC-UA server that brings together and exposes all the properties and operations from both systems.

3.3.1. Properties' integration tests

These tests (Table 8) use the Synchronous Query/Retrieve process (Figure 7) to validate the M-RobotVI data uploading process and data availability through the IMS. Retrieving times are indicative (as they vary from one request to another) and represents the average time after having performed several requests. These time measurements refer to the CoRoSect's cloud instance. The M-RobotVI set doesn't provide a simulation testbed, so all tests have been done against the real devices.

Table 8. Properties' availability tests for M-RobotVI

AAS:	SUBMODEL		Simulator	Real Device/System	
MRobotVI	Name	Element	Read	Read	Retrieving time
DATA (Proper	Values	OperationalData	VIResults	Not Supported	Checked	322 ms

		AssetConditionMonitoring	MRobotMovementStatus	Not Supported	Checked	307 ms
			MRobotStatus	Not Supported	Checked	307 ms
			MRobotObjectHeld	Not Supported	Not Implemented	
			MRobotTaskStatus	Not Supported	Checked	307 ms
			MRobotTaskID	Not Supported	Checked	307 ms
			VITaskStatus	Not Supported	Checked	307 ms
			VITaskID	Not Supported	Checked	307 ms
			VIStatus	Not Supported	Checked	307 ms
			CobotDRobotWorkspaceFree	Not Supported	Checked	307 ms
		TechnicalData	MRobotTaskConfigured	Not Supported	Checked	296 ms
			MRobotMovementStatusConfigured	Not Supported	Checked	296 ms
			VIConfiguredInspectionType	Not Supported	Checked	296 ms
			VIConfiguredFarmType	Not Supported	Checked	296 ms
			VIConfiguredDOL	Not Supported	Checked	296 ms
			StatusConfigured	Not Supported	Not Implemented	
			TaskStatusConfigured	Not Supported	Not Implemented	
		Nameplate	ManufacturerName	Not Supported	Not Implemented	
			ManufacturerProductDesignation	Not Supported	Not Implemented	
			Country code	Not Supported	Not Implemented	
			Street	Not Supported	Not Implemented	
			Zip	Not Supported	Not Implemented	
			CityTown	Not Supported	Not Implemented	
			StateCounty	Not Supported	Not Implemented	
			ManufacturerProductFamily	Not Supported	Not Implemented	
			YearOfConstruction	Not Supported	Not Implemented	
			SerialNumber	Not Supported	Checked	296 ms
			ClassificationSystem	Not Supported	Not Implemented	
			DateOfManufacture	Not Supported	Not Implemented	
			ProductCountryOfOrigin	Not Supported	Not Implemented	
			QrCode	Not Supported	Not Implemented	
			ProductIdentifier	Not Supported	Not Implemented	

3.3.2. Operations' triggering tests

These tests (Table 9 to Table 11) use the INVOKE HTTP process (Figure 9) to validate the M-Robot and its attached Visualization Module (M-RobotVI) connectivity with the Shop Floor Manager. This validates the availability of the M-Robot to be commanded by the SFM and fully integrated with the CoRoSect MES.

Table 9. List of supported commands from M-RobotVI AAS

AAS: MRobotVI	SUBMODEL Name	Element	Simulator	Real Device/System
COMMANDS (Operations)	OperationalCapabilities	MRobotExecuteTask	Not Supported	Validated
		MRobotStop	Not Supported	Implemented/Not Tested
		MRobotResume	Not Supported	Implemented/Not Tested
		MRobotReturnErrorStack	Not Supported	Implemented/Not Tested
		MRobotEmergencyStop	Not Supported	Implemented/Not Tested
		VIStart	Not Supported	Validated
		VIStop	Not Supported	Implemented/Not Tested
		VIReturnErrorStack	Not Supported	Not Implemented
		VIHistData	Not Supported	Not Implemented

Table 10. MRobotExecuteTask command test for M-RobotVI

COMMANDS (Operations)	Operational Capability	**Submodel Element**		**MRobotExecuteTask**		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**MRobotVI**/submodels/**OperationalCapability**/submodel/submodelElements/**MRobotExecuteTask**/invoke		
			InputVariable	{"idShort":"TaskID", "value":1, "descriptions":[{"language":"en", "text":"To give the task id to be executed"}]}		
			Output		**Simulator**	**Real Device/System**
		Results		"operationResult": { "success": true, "isException": false }	Checked	Checked
			Properties updated	AssetConditionMonitoring. MRobotTaskID	Checked	ID of the Op (105)
				AssetConditionMonitoring.MRobotTaskStatus	Checked	Status: (1 [Exec])- 2 [Success] – 3 [Fail]
				AssetConditionMonitoring.MRobotStatus	Checked	Status: (1 [Exec])- 2 [Success] – 3 [Fail]
				AssetConditionMonitoring. MRobotMovementStatus	Checked	0 (moving) 1(stopped) 2(emergency stop)
				AssetConditionMonitoring. CobotDRobotWorkspaceFree	Checked	0/1 (in/out manipulation zone)
		Errors	Description	Command's failure (Robot failure)	Checked	Checked
			Output Result	"operationResult": { "success": false, "isException": true }		

Table 11. VIStart command test for M-RobotVI

COMMANDS	Operational	**Submodel Element**	**VIStart**	
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**MRobotVI**/submodels/**OperationalCapability**/submodel/submodelElements/**VIStart**/invoke
			InputVariable	[{ "idShort":"InspectionType", "value":0,

<table>
<tr><td></td><td></td><td></td><td></td><td>"descriptions":[{
"language":"en",
"text":"Determines which inspections should be executed"}},
{
"idShort":"Farm",
"value":0,
"descriptions":[{
"language":"en",
"text":"Specify which farm`s crate"}]},
{
"idShort":"CrateID",
"value":0,
"descriptions":[{
"language":"en",
"text":"Specify which unique crate in particular"}]},
{
"idShort":"DOL",
"value":0,
"descriptions":[{
"language":"en",
"text":"Day of living of the insects"}]},
{
"idShort":"DateTime",
"value":" ",
"descriptions":[{
"language":"en",
"text":"Alignment with systems [MES] datetime"
}]
}]</td></tr>
<tr><td></td><td></td><td></td><td colspan="2">Output</td><td>Simulator</td><td>Real Device/System</td></tr>
<tr><td></td><td></td><td rowspan="5">Results</td><td colspan="2">"operationResult": {
"success": true,
"isException": false
}</td><td>Checked</td><td>Checked</td></tr>
<tr><td></td><td rowspan="4">Properties updated</td><td>AssetConditionMonitoring.VITaskStatus</td><td>Checked</td><td>0 Start – 1 Stop</td></tr>
<tr><td></td><td>AssetConditionMonitoring.VITaskID</td><td>Checked</td><td>ID numbering the Insp. Action</td></tr>
<tr><td></td><td>AssetConditionMonitoring.VIStatus</td><td>Checked</td><td>0-1-2-3</td></tr>
<tr><td></td><td>OperationalData.VIResults</td><td>Checked</td><td>Visual Insp. Results</td></tr>
<tr><td></td><td></td><td rowspan="2">Errors</td><td>Description</td><td>Command’s failure (Robot failure)</td><td rowspan="2">Checked</td><td rowspan="2">Checked</td></tr>
<tr><td></td><td></td><td>Output Result</td><td>"operationResult": {
"success": false,
"isException": true
}</td></tr>
</table>

3.4. Intelligent Crates (I-Crates)

The OAMK Intelligent Crates (I-Crate) (Figure 12) is composed by a set of sensors (temperature, humidity, PH, NH3, CO2, moisture) collecting and sending data of the insects’ crate where they’re attached to. This is further detailed in D9.2.

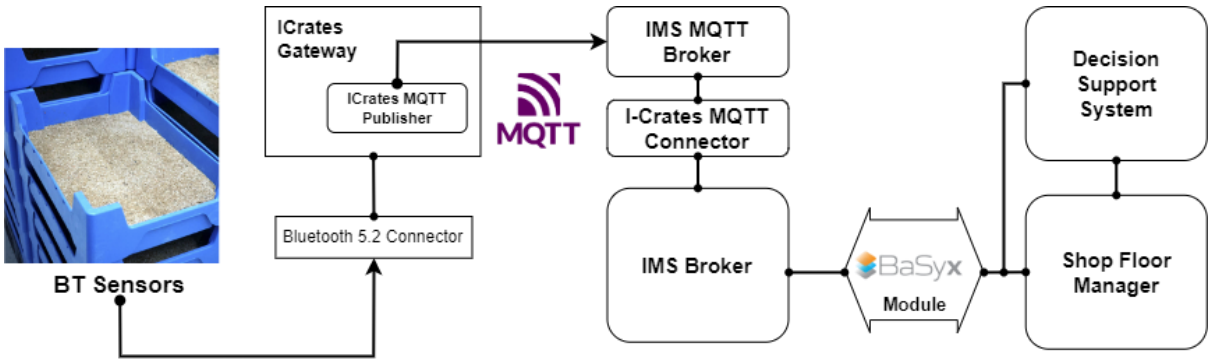

Figure 12. I-Crate integration scenario

The I-Crate works using MQTT protocol to send all sensors' data, plus its context (location, description information, etc.) to the CoRoSect's IMS. Each I-Crate uses a BT 5.2 module to connect to the crates Gateway. This GW gathers the data from all i-Crates and publishes it on the IMS MQTT Broker. These devices ONLY collect and update information, so no commands' flows are implemented.

3.4.1. Properties' integration tests

These tests (Table 12) use the Synchronous Query/Retrieve process (Figure 7) to validate the I-Crate data uploading process and data availability through the IMS. These tests are done with the OAMK's prototype connected to the CoRoSect's cloud instance. This prototype will be later replicated in the corresponding pilots' farms. Retrieving times are indicative (as they vary from one request to another) and represents the average time after having performed several requests. These times refer to the CoRoSect's cloud instance.

Table 12. Properties' availability tests for I-Crate device

AAS:		SUBMODEL		Simulator	Real Device/System	
iCrate		Name	Element	Read	Read	Retrieving time
DATA (Properties)	Values	OperationalData	TemperatureMeasure	Checked	Checked	311 ms
			TemperatureTimestamp	Checked	Checked	311 ms
			HumidityMeasure	Checked	Checked	311 ms
			HumidityTimestamp	Checked	Checked	311 ms
			CO2Measure	Checked	Checked	311 ms
			CO2Timestamp	Checked	Checked	311 ms
			MoistureMeasure	Checked	Checked	311 ms
			MoistureTimestamp	Checked	Checked	311 ms
			NH3Measure	Checked	Checked	311 ms
			NH3Timestamp	Checked	Checked	311 ms
			PHMeasure	Checked	Checked	311 ms
			PHTimestamp	Checked	Checked	311 ms
			ICrateLocation	Checked	Checked	311 ms
			TemperatureSensorLocation	Checked	Checked	311 ms
			HumiditySensorLocation	Checked	Checked	311 ms
			CO2SensorLocation	Checked	Checked	311 ms
			MoistureSensorLocation	Checked	Checked	311 ms

			NH3SensorLocation	Checked	Checked	311 ms
			PHSensorLocation	Checked	Checked	311 ms
		AssetConditionMonitoring	TemperatureSensorStatus	Checked	Not Implemented	
			HumiditySensorStatus	Checked	Not Implemented	
			CO2SensorStatus	Checked	Not Implemented	
			MoistureSensorStatus	Checked	Not Implemented	
			NH3SensorStatus	Checked	Not Implemented	
			PHSensorStatus	Checked	Not Implemented	
			TemperatureSensorBatteryLevel	Checked	Not Implemented	
			HumiditySensorBatteryLevel	Checked	Not Implemented	
			CO2SensorBatteryLevel	Checked	Not Implemented	
			MoistureSensorBatteryLevel	Checked	Not Implemented	
			NH3SensorBatteryLevel	Checked	Not Implemented	
			PHSensorBatteryLevel	Checked	Not Implemented	
			TemperatureSensorTransmissionReliability	Checked	Not Implemented	
			HumiditySensorTransmissionReliability	Checked	Not Implemented	
			CO2SensorTransmissionReliability	Checked	Not Implemented	
			MoistureSensorTransmissionReliability	Checked	Not Implemented	
			NH3SensorTransmissionReliability	Checked	Not Implemented	
			PHSensorTransmissionReliability	Checked	Not Implemented	
		Nameplate	ManufacturerName	Checked	Not Implemented	
			ManufacturerProductDesignation	Checked	Not Implemented	
			Country code	Checked	Not Implemented	
			Street	Checked	Not Implemented	
			Zip	Checked	Not Implemented	
			CityTown	Checked	Not Implemented	
			StateCounty	Checked	Not Implemented	
			ManufacturerProductFamily	Checked	Not Implemented	
			YearOfConstruction	Checked	Not Implemented	
			SerialNumber	Checked	Checked	443 ms
			ClassificationSystem	Checked	Not Implemented	
			DateOfManufacture	Checked	Not Implemented	

			ProductCountryOfOrigin	Checked	Not Implemented	
			QrCode	Checked	Not Implemented	
			ProductIdentifier	Checked	Checked	336 ms
		BillOfMaterial	TemperatureSensorSensorIdentifiers	Not Supported	Not Implemented	
			HumiditySensorSensorIdentifiers	Not Supported	Not Implemented	
			CO2SensorSensorIdentifiers	Not Supported	Not Implemented	
			MoistureSensorSensorIdentifiers	Not Supported	Not Implemented	
			NH3SensorSensorIdentifiers	Not Supported	Not Implemented	
			PHSensorSensorIdentifiers	Not Supported	Not Implemented	
	Ranges	TechnicalData	TemperatureSensorRange	Not Supported	Not Implemented	
			HumiditySensorRange	Not Supported	Not Implemented	
			CO2SensorRange	Not Supported	Not Implemented	
			MoistureSensorRange	Not Supported	Not Implemented	
			NH3SensorRange	Not Supported	Not Implemented	
			PHSensorRange	Not Supported	Not Implemented	
			TemperatureSensorMeasurementInterval	Not Supported	Not Implemented	
			HumiditySensorMeasurementInterval	Not Supported	Not Implemented	
			CO2SensorMeasurementInterval	Not Supported	Not Implemented	
			MoistureSensorMeasurementInterval	Not Supported	Not Implemented	
			NH3SensorMeasurementInterval	Not Supported	Not Implemented	
			PHSensorMeasurementInterval	Not Supported	Not Implemented	

3.5. Automated Guided Vehicle (AGV)

The Automated Guided Vehicle (AGV) system (Figure 13) includes the vehicle itself plus the embedded module which implements VDA5050 wireless protocol over WiFi to share information and receive commands. This is further detailed in D9.2.

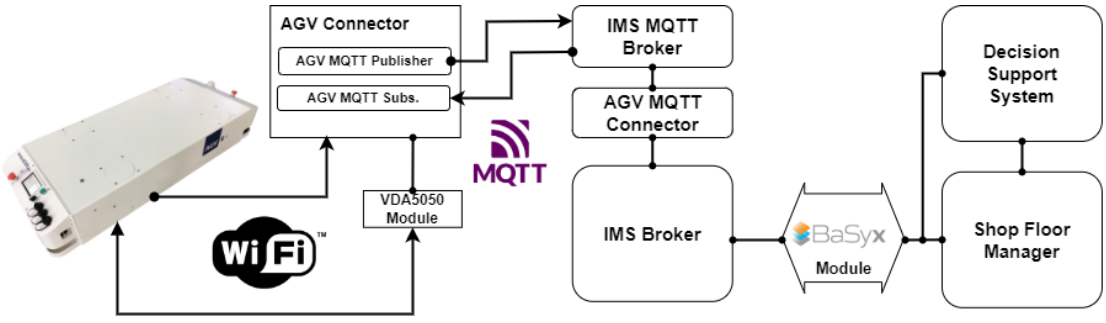

Figure 13. AGV integration with IMS scenario

The AGV system uses the IMS MQTT interface to communicate with CoRoSect's MES. Same interface (the IMS MQTT Broker) is used by the AGV for direct communications with the Route Manager and the Objects' detector.

3.5.1. Properties' integration tests

These tests (Table 13) use the Synchronous Query/Retrieve process (Figure 7) to validate the AGV data uploading process and data availability through the IMS. Retrieving times are indicative (as they vary from one request to another) and represents the average time after having performed several requests. These times refer to the CoRoSect's cloud instance.

Table 13. Properties' availability tests for AGV device

AAS:		SUBMODEL		Simulator	Real Device/System	
AGV		Name	Element	Read	Read	Retrieving time
DATA (Properties)	Values	AssetConditionMonitoring	Status	Checked	Checked	336 ms
			ConnectionState	Checked	Not Implemented	
			BatteryState	Checked	Not Implemented	
			BatteryCharge	Checked	Not Implemented	
			BatteryVoltage	Checked	Checked	336 ms
			BatteryHealth	Checked	Checked	336 ms
			Charging	Checked	Checked	336 ms
			Reach	Not Supported	Not Implemented	
			OperatingMode	Checked	Checked	336 ms
			Errors	Checked	Not Implemented	
			ErrorType	Checked	Not Implemented	
			ErrorReferences	Checked	Not Implemented	
			Driving	Checked	Checked	336 ms
			Paused	Checked	Checked	336 ms
			SafetyState	Checked	Not Implemented	
			Estop	Checked	Not Implemented	
			FieldViolation	Checked	Checked	396 ms
		TechnicalData	ActionStateConfigured	Checked	Checked	396 ms
			OperatingModeConfigured	Checked	Checked	396 ms
			StatusConfigured	Not Supported	Not Implemented	
			EStopConfigured	Not Supported	Not Implemented	
			ConnectionStateConfigured	Not Supported	Not Implemented	
		Nameplate	ManufacturerName	Checked	Not Implemented	
			ManufacturerProductDesignation	Checked	Not Implemented	
			Country code	Checked	Not Implemented	
			Street	Checked	Not Implemented	
			Zip	Checked	Not Implemented	

		CityTown	Checked	Not Implemented	
		StateCounty	Checked	Not Implemented	
		ManufacturerProductFamily	Checked	Not Implemented	
		YearOfConstruction	Checked	Not Implemented	
		SerialNumber	Checked	Not Implemented	
		ClassificationSystem	Checked	Not Implemented	
		DateOfManufacture	Checked	Not Implemented	
		ProductCountryOfOrigin	Checked	Not Implemented	
		QrCode	Checked	Not Implemented	
		ProductIdentifier	Checked	Not Implemented	
	OperationalData	OrderId	Checked	Checked	406 ms
		OrderUpdateId	Checked	Checked	406 ms
		ZoneSetId	Checked	Checked	406 ms
		LastNodeId	Checked	Checked	406 ms
		LastNodeSequenceId	Checked	Checked	406 ms
		NodeStates	Checked	Not Implemented	
		NodeId	Checked	Not Implemented	
		NodeSequenceId	Checked	Not Implemented	
		NodeDescription	Checked	Not Implemented	
		NodePosition	Checked	Not Implemented	
		x_AGVPosition	Checked	Checked	406 ms
		y_AGVPosition	Checked	Checked	406 ms
		Theta_AGVPosition	Checked	Checked	406 ms
		AllowedDeviationXY	Checked	Not Implemented	
		AllowedDeviationTheta	Checked	Not Implemented	
		Nodereleased	Checked	Not Implemented	
		EdgeStates	Not Supported	Not Implemented	
		EdgeId	Not Supported	Not Implemented	
		EdgesequenceId	Not Supported	Not Implemented	
		EdgeDescription	Not Supported	Not Implemented	
		Edgereleased	Not Supported	Not Implemented	
		Trajectory	Not Supported	Not Implemented	
		Degree	Not Supported	Not Implemented	
		KnotVector	Not Supported	Not Implemented	
		ControlPoints	Not Supported	Not Implemented	

			AgvPosition	Not Supported	Not Implemented	
			PositionInitialized	Not Supported	Not Implemented	
			LocalizationScore	Checked	Checked	406 ms
			DeviationRange	Not Supported	Not Implemented	
			MapId	Checked	Checked	406 ms
			MapDesc	Checked	Checked	406 ms
			Velocity	Not Supported	Not Implemented	
			Vx	Checked	Checked	406 ms
			Vy	Checked	Checked	406 ms
			Omega	Checked	Checked	406 ms
			Loads	Checked	Not Implemented	
			LoadId	Checked	Checked	406 ms
			LoadType	Checked	Checked	406 ms
			LoadPosition	Checked	Not Implemented	
			BoundingBoxReference	Not Supported	Not Implemented	
			LoadDimensions	Not Supported	Not Implemented	
			Length	Not Supported	Not Implemented	
			Width	Not Supported	Not Implemented	
			Height	Not Supported	Not Implemented	
			Weight	Not Supported	Not Implemented	
			NewBaseRequest	Checked	Checked	406 ms
			DistanceSinceLastNode	Checked	Checked	406 ms
			ActionStates	Checked	Not Implemented	
			Actionid	Checked	Not Implemented	
			ActionType	Checked	Not Implemented	
			ActionDescription	Checked	Not Implemented	
			Drop_ActionStatus	Checked	Checked	406 ms
			Pick_ResultDescription	Checked	Checked	406 ms
			Drop_ResultDescription	Checked	Checked	406 ms
			Pick_ActionStatus	Checked	Checked	406 ms
			StopPause_ActionStatus	Checked	Checked	406 ms
			CancelOrder_ActionStatus	Checked	Checked	406 ms
			StopPause_ResultDescription	Checked	Checked	406 ms
			StartPause_ResultDescription	Checked	Checked	406 ms
			StartPause_ActionStatus	Checked	Checked	406 ms
			CancelOrder_ResultDescription	Checked	Checked	406 ms

3.5.2. Operations' triggering tests

These tests (Table 14 to Table 19) use the INVOKE HTTP process (Figure 9) to validate the AGV connectivity with the Shop Floor Manager. This validates the availability of the AGV to be commanded by the SFM and fully integrated with the CoRoSect MES.

Table 14. List of supported commands for AGV AAS

AAS: AGV1	SUBMODEL Name	Element	Simulator	Real Device/System
COMMANDS (Operations)	OperationalCapabilities	StartPause	Supported	Validated
		StopPause	Supported	Validated
		StartCharging	Supported	Not Implemented
		StopCharging	Supported	Not Implemented
		InitPosition	Supported	Not Implemented
		StateRequest	Not Supported	Not Implemented
		LogReport	Not Supported	Not Implemented
		Pick	Supported	Validated
		Drop	Supported	Validated
		DetectObject	Not Supported	Not Implemented
		FinePositioning	Not Supported	Not Implemented
		WaitForTrigger	Not Supported	Not Implemented
		CancelOrder	Supported	Validated

Table 15. StartPause command test for AGV

COMMANDS (Operations)	Operational Capability	**Submodel Element**		**StartPause**	
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**AGV1**/submodels/**OperationalCapability**/submodel/submodelElements/**StartPause**/invoke	
			InputVariable	none	
			Output	**Simulator**	**Real Device/System**
		Results	{ "idShort": "CommandStatus", "value": "", "descriptions": [{ "language": "en", "text": "Status of the output of start paused" }]}	Checked	Checked
			Properties updated: OperationalData.StartPause_ActionStatus	Checked	0 (Waiting); 1 (initializing); 2 (running); 3 (paused); 4 (finished); 5 (failed)
		Errors	Description: Command's failure (AGV failure)	Checked	Checked
			Output Result: "operationResult": { "success": false, "isException": true}		

Table 16. StopPause command test for AGV

COMMANDS (Operations)	Operational Capability	Submodel Element		StopPause		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**AGV1**/submodels/**OperationalCapability**/submodel/submodelElements/**StopPause**/invoke		
			InputVariable	none		
			Output		**Simulator**	**Real Device/System**
		Results	{ "idShort": "CommandStatus", "value": "", "descriptions": [{ "language": "en", "text": "Status of the output of stop paused" }]}		Checked	Checked
			Properties updated	OperationalData.StopPause_ActionStatus	Checked	0 (Waiting); 1 (initializing); 2 (running); 3 (paused); 4 (finished); 5 (failed)
		Errors	Description	Command's failure (AGV failure)	Checked	Checked
			Output Result	"operationResult": { "success": false, "isException": true}		

Table 17. Pick command test for AGV

COMMANDS (Operations)	Operational Capability	Submodel Element		pick		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**AGV1**/submodels/**OperationalCapability**/submodel/submodelElements/**pick**/invoke		
			InputVariable	[{"idShort":"loadType","value":"EURO"}, {"idShort":"loadId","value":"1"}, {"idShort":"height","value":"0"}, {"idShort":"depth","value":"0"}, {"idShort":"side","value":"0"}]		
			Output		**Simulator**	**Real Device/System**
		Results	{"idShort": "CommandStatus", "descriptions": [{"language": "en", "text": "Status of the output of pick" }],"value": "1"}		Checked	Checked
			Properties updated	OperationalData.Pick_ActionStatus	Checked	0 (Waiting); 1 (initializing); 2 (running); 3 (paused); 4 (finished); 5 (failed)
		Errors	Description	Command's failure (AGV failure)	Checked	Checked
			Output Results	"operationResult": { "success": false, "isException": true}		

Table 18. Drop command test for AGV

COMMANDS (Operations)	Operational Capability	**Submodel Element**		**drop**		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**AGV1**/submodels/**OperationalCapability**/submodel/submodelElements/**drop**/invoke		
			InputVariable	[{"idShort":"loadType","value":"EURO"}, {"idShort":"loadId","value":"1"}, {"idShort":"height","value":"0"}, {"idShort":"depth","value":"0"}, {"idShort":"side","value":"0"}]		
			Output		**Simulator**	**Real Device/System**
		Results		{"idShort": "CommandStatus", "descriptions": [{"language": "en", "text": "Status of the output of drop" }],"value": "1"}	Checked	Checked
			Properties updated	OperationalData.Drop_ActionStatus	Checked	0 (Waiting); 1 (initializing); 2 (running); 3 (paused); 4 (finished); 5 (failed)
		Errors	Description	Command's failure (AGV failure)	Checked	Checked
			Output Results	"operationResult": { "success": false, "isException": true}		

Table 19. CancelOrder command test for AGV

COMMANDS (Operations)	Operational Capability	**Submodel Element**		**CancelOrder**		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**AGV1**/submodels/**OperationalCapability**/submodel/submodelElements/**CancelOrder**/invoke		
			InputVariable	none		
			Output		**Simulator**	**Real Device/System**
		Results		{"idShort": "CommandStatus", "descriptions": [{"language": "en", "text": "Status of the output of Cancel Order" }],"value": "1"}	Checked	Checked
			Properties updated	OperationalData.CancelOrder_ActionStatus	Checked	0 (Waiting); 1 (initializing); 2 (running); 3 (paused); 4 (finished); 5 (failed)
				OperationalData.Pick/Drop_ActionStatus	Checked	From 2 (running) to 4 (finished)
				OperationalData.StarPause/StopPause_ActionStatus	Checked	From 2 (running) to 4 (finished)
		Errors	Description	Command's failure (AGV failure)	Checked	Checked
			Output Results	"operationResult": { "success": false, "isException": true}		

3.6. HoloLens System

Microsoft's HoloLens 2 (HoloLens) provides a see-through-based augmented reality system used within the human-robot collaboration scenario. This (Figure 14) includes the Microsoft device plus a specifically designed OPC interface to integrate it with the IMS. This is further detailed in D9.2.

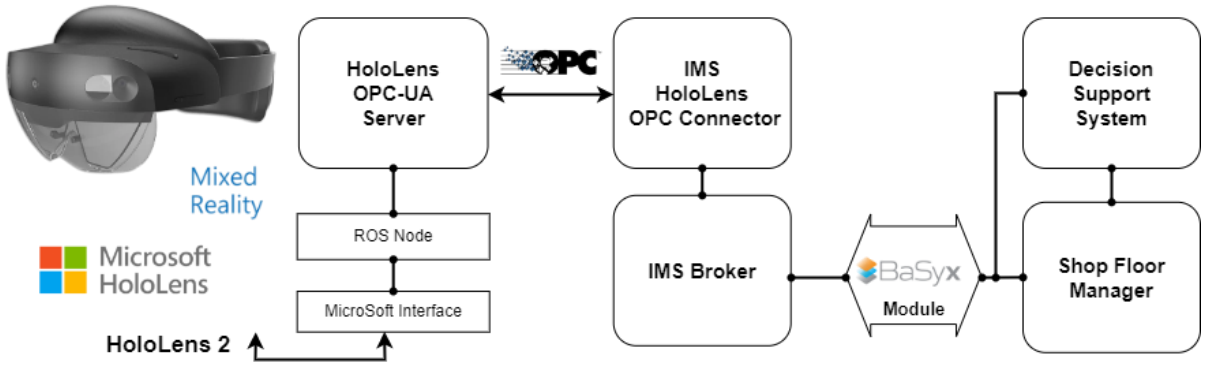

Figure 14. HoloLens' device integration scenario

3.6.1. Properties' integration tests

These tests (Table 20) use the Synchronous Query/Retrieve process (Figure 7) to validate the HoloLens device data uploading process and data availability through the IMS. The HoloLens does not provide a simulator for data retrieving, so all reading tests are done using the real device. Retrieving times are indicative (as they vary from one request to another) and represents the average time after having performed several requests. These times refer to the CoRoSect's cloud instance.

Table 20. Properties' availability tests for HoloLens device

AAS:		SUBMODEL		Simulator	Real Device/System	
HOLOLENS		Name	Element	Read	Read	Retrieving time
DATA (Properties)	Values	AssetConditionMonitoring	Status	Not Supported	Not Implemented	
			DisplayStopped	Not Supported	Checked	226 ms
		Technical Data	ConfiguredAssets	Not Supported	Checked	226 ms
			StatusConfiguration	Not Supported	Checked	226 ms
			MessagesConfiguration	Not Supported	Checked	226 ms
		Nameplate	ManufacturerName	Not Supported	Checked	293 ms
			ManufacturerProductDesignation	Not Supported	Checked	293 ms
			Country code	Not Supported	Checked	293 ms
			Street	Not Supported	Checked	293 ms
			Zip	Not Supported	Checked	293 ms
			CityTown	Not Supported	Checked	293 ms
			StateCounty	Not Supported	Checked	293 ms
			ManufacturerProductFamily	Not Supported	Checked	293 ms
			YearOfConstruction	Not Supported	Checked	293 ms
			SerialNumber	Not Supported	Checked	293 ms
			ClassificationSystem	Not Supported	Checked	293 ms

			DateOfManufacture	Not Supported	Checked	293 ms
			ProductCountryOfOrigin	Not Supported	Checked	293 ms
			QrCode	Not Supported	Not Implemented	
			ProductIdentifier	Not Supported	Checked	293 ms

3.6.2. Operations' triggering tests

These tests (Table 21 to Table 25) use the INVOKE HTTP process (Figure 9) to validate the HoloLens connectivity with the Shop Floor Manager. This validates the availability of the HoloLens to interact with the SFM and fully integrated with the CoRoSect MES.

Table 21. List of supported commands for Hololens' AAS

AAS: HOLOLENS	SUBMODEL Name	Element	Simulator	Real Device/System
COMMANDS (Operations)	OperationalCapabilities	DisplayMessage	Supported	Validated
		DisplayTrajectory	Supported	Validated
		StopDisplayMessage	Supported	Validated
		StopDisplayTrajectory	Supported	Validated
		ReturnStackError	Not Supported	Not Implemented

Table 22. DisplayMessage command test for HoloLens

COMMANDS (Operations)	Operational Capability	**Submodel Element**		**DisplayMessage**		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**Hololens**/submodels/**OperationalCapability** /submodel/submodelElements/**DisplayMessage**/invoke		
			InputVariable	"inputVariable": [{"idShort": "AssetID","value": 1}, {"idShort": "MessageID","value": 1}]		
			Output		**Simulator**	**Real Device/System**
		Results		"operationResult": { "success": true, "isException": false}, "outputVariable": [{"value": true}]	Checked	Checked
			Properties updated	AssetConditioningMonitoring. DisplayStopped	Checked	True –> False
		Errors	Description	Command's failure (HoloLens failure)	Checked	Checked
			Output Results	"operationResult": { "success": false, "isException": true}		

Table 23. DisplayTrajectory command test for HoloLens

COMMA	Operatio	**Submodel Element**	**DisplayTrajectory**	
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**Hololens**/submodels/**OperationalCapability** /submodel/submodelElements/**DisplayTrajectory**/invoke

			InputVariable	"inputVariable": [{"idShort": "AssetID","value": 1}, {"idShort": "TrajectoryID","value": 1}]		
			Output		**Simulator**	**Real Device/System**
		Results	"operationResult": { "success": true, "isException": false}, "outputVariable": [{"value": true}]		Checked	Checked
			Properties updated	AssetConditioningMonitoring. DisplayStopped	Checked	True –> False
		Errors	Description	Command's failure (HoloLens failure)	Checked	Checked
			Output Results	"operationResult": { "success": false, "isException": true}		

Table 24. StopDisplayMessage command test for HoloLens

COMMANDS (Operations)	**Operational Capability**	**Submodel Element**		**StopDisplayMessage**		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**Hololens**/submodels/**OperationalCapability** /submodel/submodelElements/**StopDisplayMessage**/invoke		
			InputVariable	none		
			Output		**Simulator**	**Real Device/System**
		Results	"operationResult": { "success": true, "isException": false}, "outputVariable": [{"value": true}]		Checked	Checked
			Properties updated	AssetConditioningMonitoring. DisplayStopped	Checked	False –> True
		Errors	Description	Command's failure (HoloLens failure)	Checked	Checked
			Output Results	"operationResult": { "success": false, "isException": true}		

Table 25. StopDisplayTrajectory command test for HoloLens

COMMANDS (Operations)	**Operational Capability**	**Submodel Element**		**StopDisplayTrajectory**		
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**Hololens**/submodels/**OperationalCapability** /submodel/submodelElements/**StopDisplayTrajectory**/invoke		
			InputVariable	none		
			Output		**Simulator**	**Real Device/System**
		Results	"operationResult": { "success": true, "isException": false}, "outputVariable": [{"value": true}]		Checked	Checked

			Properties updated	AssetConditioningMonitoring. DisplayStopped	Checked	False –> True
		Errors	Description	Command's failure (HoloLens failure)	Checked	Checked
			Output Results	"operationResult": { "success": false, "isException": true}		

3.7. Route Manager

The Route Manager (RM) software (Figure 15) is deployed in a dedicated server with GPU and uses the MQTT interface to interact with the IMS and with the AGV using VDA5050. This is further detailed in D9.2.

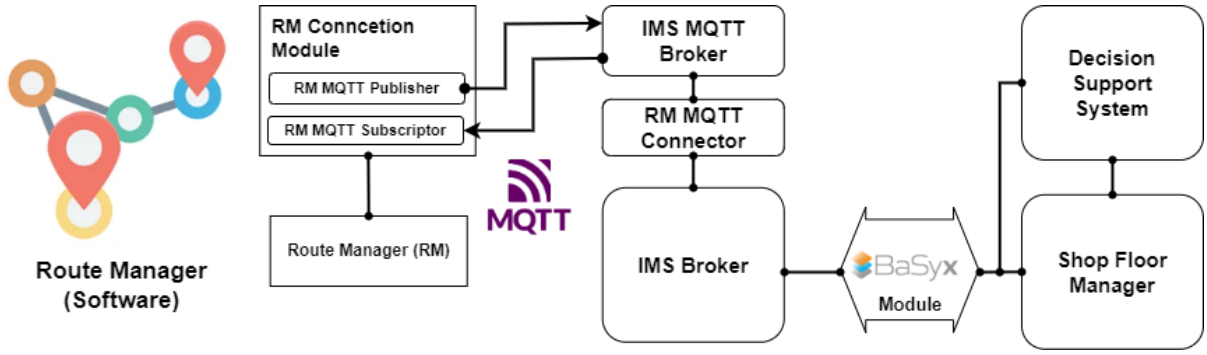

Figure 15. Route Manager and MES integration scenario

3.7.1. Properties' integration tests

These tests (Table 26) use the Synchronous Query/Retrieve process (Figure 7) to validate the Route Manager data uploading process and data availability through the IMS. Retrieving times are indicative (as they vary from one request to another) and represents the average time after having performed several requests. These times refer to the CoRoSect's cloud instance.

Table 26. Properties' availability tests for Route Manager device

AAS:		SUBMODEL		Simulator	Real Device/System	
RouteManager		Name	Element	Read	Read	Retrieving time
DATA (Properties)	Values	AssetConditionMonitoring	Status	Checked	Checked	319 ms
			ErrorMessage	Checked	Checked	319 ms
			InformationSourceforStatus	Checked	Checked	319 ms
			TimeofStatusChange	Checked	Checked	319 ms
		Technical Data	ActionStateConfigured	Checked	Checked	205 ms
		Nameplate	ManufacturerName	Checked	Checked	275 ms
			ManufacturerProductDesignation	Checked	Checked	275 ms
			Country code	Checked	Checked	275 ms
			Street	Checked	Checked	275 ms
			Zip	Checked	Checked	275 ms

			CityTown	Checked	Checked	275 ms
			StateCounty	Checked	Checked	275 ms
			ManufacturerProductFamily	Checked	Checked	275 ms
			YearOfConstruction	Checked	Checked	275 ms
			SerialNumber	Checked	Checked	275 ms
			ClassificationSystem	Checked	Checked	275 ms
			DateOfManufacture	Checked	Checked	275 ms
			ProductCountryOfOrigin	Checked	Checked	275 ms
			QrCode	Checked	Checked	275 ms
			ProductIdentifier	Checked	Checked	275 ms
		Operational Data	MapPetition	Checked	Checked	266 ms
			RouteStatus	Checked	Checked	266 ms

3.7.2. Operations' triggering tests

These tests (Table 27 to Table 30) use the INVOKE HTTP process (Figure 9) to validate the Route Manager connectivity with the Shop Floor Manager. This validates the availability of the Route Manager to interact with the SFM and be fully integrated with the CoRoSect MES.

Note: The Route Manager uses MQTT protocol for both: operations and properties. This MQTT implementation works in a different way that OPC does and the responses (routes) to the invoked commands are provided within the returned payloads of the responses (OperationResults) and not through device's properties.

Table 27. List of supported commands for Route Manager AAS

AAS: RouteManager	SUBMODEL Name	Element	Simulator	Real Device/System
COMMANDS (Operations)	Operational Capabilities	NewRoute	Supported	Supported
		CancelRoute	Supported	Supported
		StartRoute	Supported	Supported

Table 28. NewRoute command test for RouteManager

COMMANDS (Operations)	Operational Capability	**Submodel Element**		**NewRoute**	
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**RouteManager**/submodels/**OperationalCapability**/submodel/submodelElements/**NewRoute**/invoke	
			InputVariable	none	
			Output	**Simulator**	**Real Device/System**
		Results	[{"idShort": "NewRouteStatus", "value": "Valid route from [point55] to [point4] for [AGV1]."}, {"idShort": "uuId", "value": "c397a78c-d9a3-49cb-82e3-015489f8e9d7"},	Checked	Checked

			{"idShort": "TimeStamp", "value": "2023-05-30T15:59:24.616148Z"}, {"idShort": "rId", "value": "b0586173-362c-4846-92c3-c6b380e254d8"}, {"idShort": "AGVId", "value": "AGV1"}]		

Table 29. CancelRoute command test for RouteManager

COMMANDS (Operations)	**Operational Capability**	**Submodel Element**		**CancelRoute**	
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**RouteManager**/submodels/**OperationalCapability**/submodel/submodelElements/**CancelRoute**/invoke	
			InputVariable	{"idShort":"rId","value":"all"}	
			Output	**Simulator**	**Real Device/System**
		Results	[{"idShort": "CancelStatus", "value":"Route rid:all has been canceled"}, {"idShort": "uuId", "value": "c397a78c-d9a3-49cb-82e3-015489f8e9d7"}, {"idShort": "TimeStamp", "value": "2023-05-30T15:59:24.616148Z"}, {"idShort": "rId", "value": "b0586173-362c-4846-92c3-c6b380e254d8"}, {"idShort": "AGVId", "value": "AGV1"}]	Checked	Checked

Table 30. StartRoute command test for RouteManager

COMMANDS (Operations)	**Operational Capability**	**Submodel Element**		**StartRoute**	
		INVOKE operation	*REST call (POST)*	[BaSyx_Server]/aas/**RouteManager**/submodels/**OperationalCapability**/submodel/submodelElements/**StartRoute**/invoke	
			InputVariable	none	
			Output	**Simulator**	**Real Device/System**
		Results	[{"idShort": "StartStatus", "value":"Route id b71f4d45-03e6-4b96-9ce2-a85d371172b4 for the AGV AGV1 has started.""}, {"idShort": "uuId", "value": "c397a78c-d9a3-49cb-82e3-015489f8e9d7"}, {"idShort": "TimeStamp", "value": "2023-05-30T15:59:24.616148Z"}, {"idShort": "rId", "value": "b0586173-362c-4846-92c3-c6b380e254d8"}, {"idShort": "AGVId", "value": "AGV1"}]	Checked	Checked

3.8. Objects detector

The Objects/Obstacles Detector (OD) system (Figure 16) is composed by a software module plus several fixed IP cameras, either connected by WiFi or Ethernet cable. This is further detailed in D9.2.

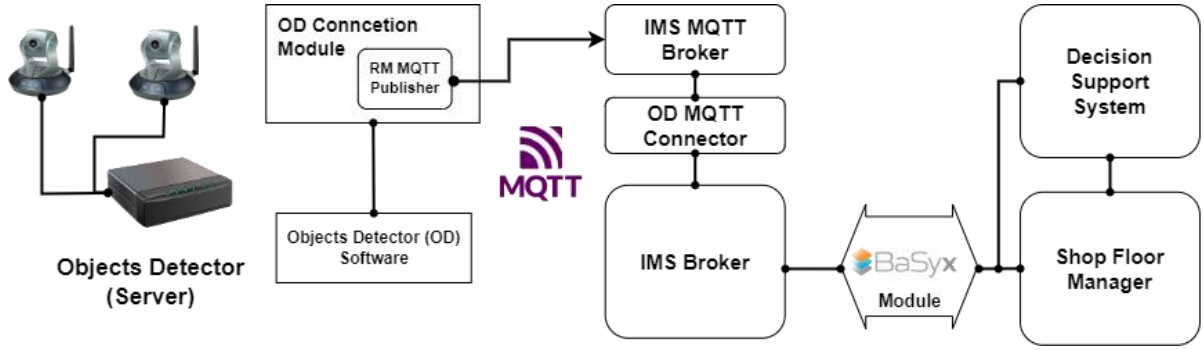

Figure 16. I-Crate integration scenario

The Objects detector system uses the IMS MQTT interface to communicate with CoRoSect's MES. Same interface (the IMS MQTT Broker) is used for direct communications with the Route Manager.

3.8.1. Properties' integration tests

These tests (Table 31) use the Synchronous Query/Retrieve process (Figure 7) to validate the OD data uploading process and data availability through the IMS. Retrieving times are indicative (as they vary from one request to another) and represents the average time after having performed several requests. These times refer to the CoRoSect's cloud instance.

Table 31. Properties' availability tests for Objects Detector system

AAS:		SUBMODEL		Simulator	Real Device/System	
Object Detector		Name	Element	Read	Read	Retrieving time
DATA (Properties)	Values	AssetCondition Monitoring	Status	Checked	Checked	289 ms
			ErrorMessage	Checked	Checked	289 ms
			InformationSourceforStatus	Checked	Checked	289 ms
			TimeofStatusChange	Checked	Checked	289 ms
		Technical Data	StatusConfigured	Checked	Checked	195 ms
		Nameplate	ManufacturerName	Checked	Checked	255 ms
			ManufacturerProductDesignation	Checked	Checked	255 ms
			Country code	Checked	Checked	255 ms
			Street	Checked	Checked	255 ms
			Zip	Checked	Checked	255 ms
			CityTown	Checked	Checked	255 ms
			StateCounty	Checked	Checked	255 ms
			ManufacturerProductFamily	Checked	Checked	255 ms
			YearOfConstruction	Checked	Checked	255 ms

			SerialNumber	Checked	Checked	255 ms
			ClassificationSystem	Checked	Checked	255 ms
			DateOfManufacture	Checked	Checked	255 ms
			ProductCountryOfOrigin	Checked	Checked	255 ms
			QrCode	Checked	Checked	255 ms
			ProductIdentifier	Checked	Checked	255 ms
		OperationalData	InformationSource	Checked	Checked	224 ms
			TimeofDetection	Checked	Checked	224 ms
			DetectionUnqiueId	Checked	Checked	224 ms
			TypeofObstacle	Checked	Checked	224 ms
			DetectedObjectPosition	Checked	Checked	224 ms
			PredictedTrajectory	Checked	Checked	224 ms
			DetectionMessage	Checked	Checked	224 ms

4. Functional tests

Besides the integration of the Shop Floor components and MES building blocks, supporting data and commands flows, the full CoRoSect System must implement specific functional requirements related to data management and distribution capabilities. The full list of requirements is shown in D2.4 but this section tests the **historical data storage** (for both, properties, and commands) and the **data distribution capabilities** (synchronous/asynchronous queries), as the rest of functionalities have been already evaluated in the previous sections.

4.1. Data gathering, storage and presentation

Section 3 has described the functionalities (APIs) and validated the interfaces used by the IMS to gather all the information from the shop floor integrated components. Using these validated Digital Twins structures (sub-models, properties, and operations), the IMS also stores the changes and the evolution of these parameters through the shop floor execution time, creating a historical data base. In this sense, the system keeps registries of:

- Any modification done in the registered AAS entities, including new registries and variations of their attributes.
- Registries and modifications done in the linked asset instances
- Registries and modifications done in the AAS lists of sub-models
- Updates on AAS properties (through the corresponding Submodel Element Properties)
- All commands (Submodel Element Operations) triggered by the Shop Floor Manager through the BaSyx interface, including the input variables
- All the direct responses (output variables in corresponding Submodel Element Operations) returned by the corresponding connectors as a response to a requested command.

These historical data storages cover two main system functionalities:

1. Allow tracking and monitoring of:
    - The evolution through time of specific properties, such as temperature and humidity of the shop floor i-crates.
    - The commands' flows during a given time slot, including the monitoring of specific status properties which reflects the corresponding tasks' evolution.
2. Create valuable datasets that can be downloaded and exploited by external ERP systems to evaluate overall shop floor processes (manufacturing processes)

A shown in Figure 17, the IMS implementation offers two different interfaces to provide the historical datasets to external ERP systems: i) a proper SQL API REST interface, which allows the execution of Simple Query Language requests; and ii) an NGSI based Time Series API REST to request properties' evolution according to time.

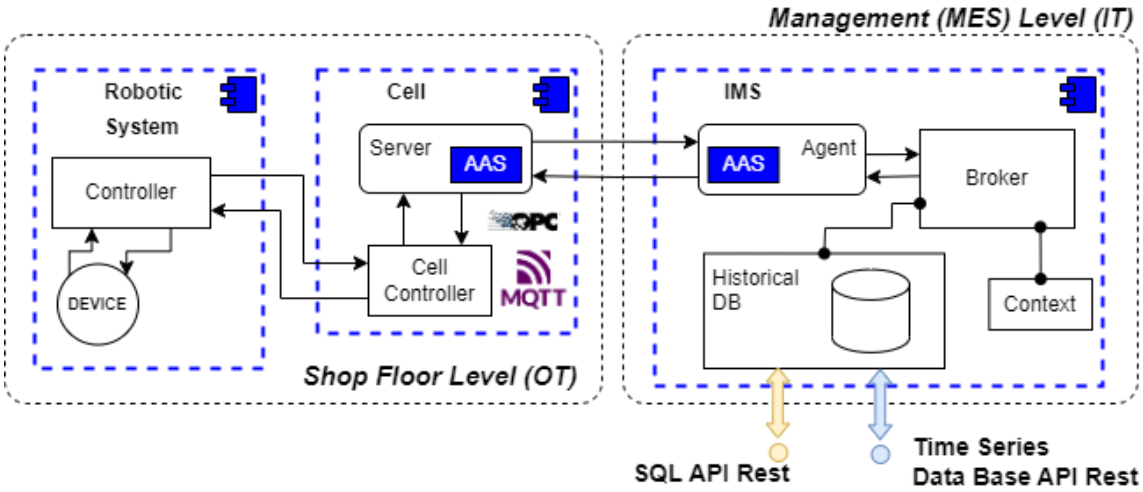

Figure 17. Historical datasets gathering and retrieval schema

4.1.1. Historical data retrieval: SQL Interface

As already introduced in D9.2 and D2.4, the IMS implements an internal Database that provides an API Rest supporting SQL queries[3]. The internal structure of tables is based, in turn, on the properties and operations that define the Digital Twins (D9.1 interfaces) and is to be detailed in D4.3.

Next SQL test queries the last 5 Temperature measures for the *hotelli_musta* I-Crate deployed in Italian Cricket Farm pilot (ICF) ordered by the time these where captured. All historical properties for ICF pilot are stored in the **mticf.eti4submodelelementproperty** table.

Execute SQL query (_sql REST API) to CoRoSect System IMS		
POST	[*CoRoSect_SQL_Server*]/**_sql**	
Headers	Fiware-Service	*No headers required. Queried DB is specified in the payload*
	Fiware-ServicePath	
Payload	JSON with SQL statement in a string.	{"stmt":"SELECT time_index as time, entity_id as ID, idshort as PARAM, Value FROM **mticf.eti4submodelelementproperty** where idshort = 'TemperatureMeasure' AND refi4aasid like '%hotelli_musta%' ORDER by time_index DESC LIMIT 5"}

Return values		
code	404	AAS and/or Submodel and/or Submodel Element Property not found
code	50X	Server error
code	200/201	Success: retrieves JSON object including columns' names and rows data retrieved


```
{
  "cols": ["time", "id", "param", "value"],
  "rows": [
    [
      1686844271883,
      "urn:ngsi-v2:RAMI40:I4SubmodelElementProperty:OperationalData:TemperatureMeasure:hotelli_musta",
      "TemperatureMeasure",
      "26.18"
    ],
    [
      1686844228319,
      "urn:ngsi-v2:RAMI40:I4SubmodelElementProperty:OperationalData:TemperatureMeasure:hotelli_musta",
      "TemperatureMeasure",
      "26.17"
    ],
    [
      1686844130453,
      "urn:ngsi-v2:RAMI40:I4SubmodelElementProperty:OperationalData:TemperatureMeasure:hotelli_musta",
```


[3] https://crate.io/docs/crate/reference/en/5.3/interfaces/http.html

```
        "TemperatureMeasure",
        "26.16"
      ],
      [
        1686844096661,
        "urn:ngsi-v2:RAMI40:I4SubmodelElementProperty:OperationalData:TemperatureMeasure:hotelli_musta",
        "TemperatureMeasure",
        "26.15"
      ],
      [
        1686844016541,
        "urn:ngsi-v2:RAMI40:I4SubmodelElementProperty:OperationalData:TemperatureMeasure:hotelli_musta",
        "TemperatureMeasure",
        "26.14"
      ]
    ],
    "rowcount": 5, "duration": 33.128746
}
```


4.1.2. Historical data retrieval: Time Series

As also presented in D9.2 and D2.4, the internal IMS Database provides an API Rest supporting NGSI Time Series queries (NGSI-TSDB)[4]. The Digital Twins' structure (D9.1 interfaces) and the list of properties and operations defines the IDs to be used for the request (these are to be detailed in D4.3).

Next NGSI-TSDB test replicates the previous SQL query requesting the last 5 Temperature measures for the hotelli_musta I-Crate deployed in Italian Cricket Farm pilot (ICF). All historical properties for ICF pilot are stored within the **icf** fiware service.

Execute TSDB query (NGSI-TSDB REST API) to CoRoSect System IMS		
GET	[*CoRoSect_TSDB_Server*]/v2/entities/**urn:ngsi-v2:RAMI40:I4SubmodelElementProperty:OperationalData:TemperatureMeasure:hotelli_musta**/attrs/**value**?lastN=5	
Headers	Fiware-Service	icf *(for ICF pilot query)*
	Fiware-ServicePath	
Payload	None	

Return values		
code	404	Property (entity and/or attribute) not found
code	50X	Server error
code	200	Success: retrieves JSON object including time index and values of the requested property.


```
{
  "attrName": "value",
  "entityId": "urn:ngsi-v2:RAMI40:I4SubmodelElementProperty:OperationalData:TemperatureMeasure:hotelli_musta",
  "entityType": "I4SubmodelElementProperty",
  "index": [
    "2023-06-15T15:46:56.541+00:00",
    "2023-06-15T15:48:16.661+00:00",
    "2023-06-15T15:48:50.453+00:00",
    "2023-06-15T15:50:28.319+00:00",
    "2023-06-15T15:51:11.883+00:00"],
  "values": [
    "26.14",
    "26.15",
    "26.16",
    "26.17",
    "26.18"]
}
```


[4] https://app.swaggerhub.com/apis/smartsdk/ngsi-tsdb/0.8.3

4.1.3. Historical data presentation dashboards

CoRoSect Integrated System on its version 1 implements an internal instance of Grafana[5], an open tool to create and customise dashboards. This tool uses the mentioned historical access APIs to define specific visualizations of the gathered data, complementing, this way, the presentation layer for the historical records gathered and stored by the MES Information Management System (IMS). Figure 18 shows the dashboard designed to monitor the I-Crates to be deployed in pilots. Specifically, the dashboard presents the status (last reported values for the monitored properties) for *hotelli_musta* I-Crate plus the evolution during the last 2 hours of these parameters.

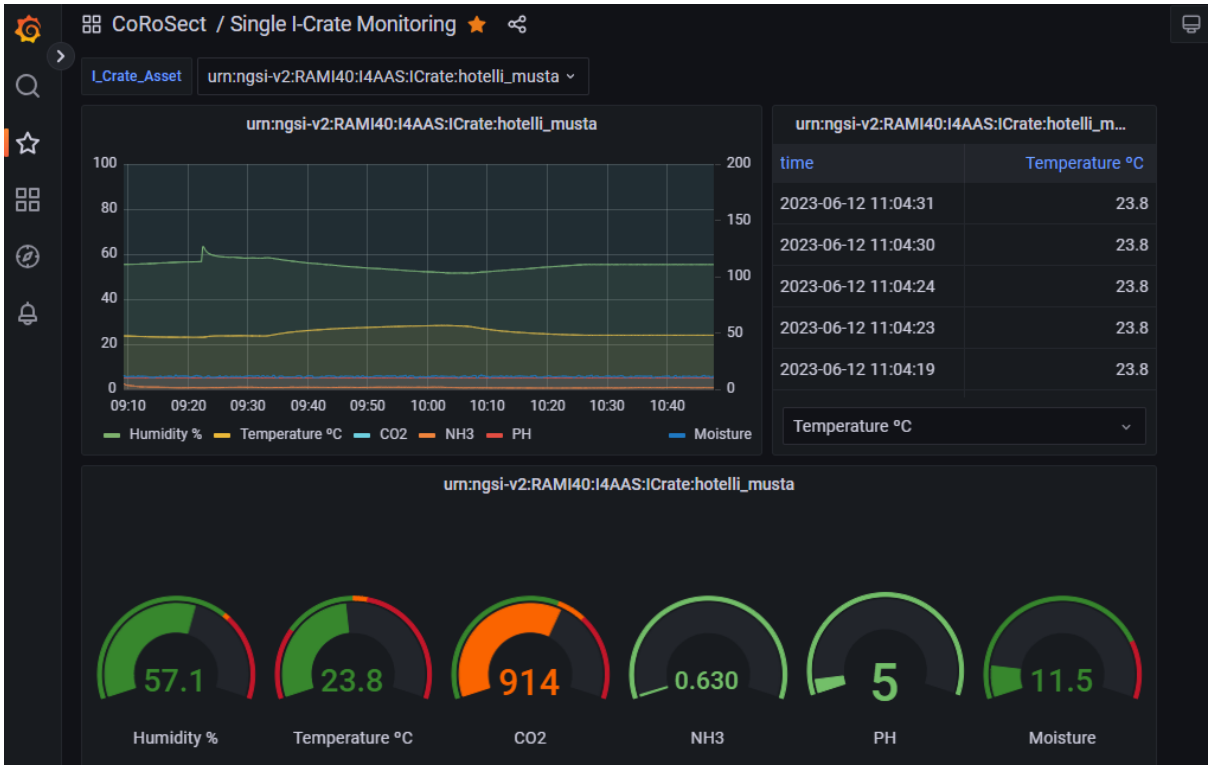

Figure 18. Dashboard test for hotelli_musta I-Crate: Parameters' monitoring.

4.2. Synchronous and Asynchronous information management

From the set of requirements listed in D2.4, the CoRoSect integrated system, and specifically its Information Management System (IMS), must support:

- The IMS provides data to other components (IT Level) upon request (synchronous data query/retrieve)
- The IMS implements publish/subscribe mechanisms to distribute data among other components – IT Level (asynchronous data query/retrieve)

These two requirements are supported, according to D9.2, using the RAMI4.0 compliant Eclipse BaSyx[6] API implementation and through the NGSIv2[7] interface.

4.2.1. Synchronous data query/retrieve

This functionality allows the SFM/DSS (or any other external ERP/Data exploitation system) to directly query the IMS and retrieve the structure and last reported values of any property and/or command

[5] https://grafana.com/
[6] https://wiki.eclipse.org/BaSyx_/_Documentation_/_API_/_AssetAdministrationShell
[7] https://fiware-orion.readthedocs.io/en/1.13.0/user/walkthrough_apiv2/

linked to any registered Digital Twin, as well as to retrieve the full Digital Twin structure and associated asset. This is supported by two implemented HTTP REST APIs:

- The NGSIv2 API REST, natively supported by the deployed CoRoSect's Context Broker,
- And a CoRoSect's customisation of the Eclipse BaSyx version 1, as an implementation of an API REST on top of the NGSI specifications.

This BaSyx interface is the one used to validate the data availability, according to section 3.1.1 and shown through the rest of section 3. The process followed to validate the shop floor DTs implementation, validates in turn the synchronous query/retrieve functionality. As an example of the validation queries done, the following request retrieves the Status value of the Stacking/Destacking Robot

- Asset Administrarion Shell ID (AAS_ID) = AASDRobot
- Sub-model ID (Submodel_ID) = AssetConditionMonitoring
- Sub-model Element ID (SubmodelElement_ID) = Status

Retrieve Submodel Elements' VALUES (for Submodel Elements' PROPERTIES)		
GET	[*BaSyx Server*]/ aas/**AASDRobot**/submodels/**AssetConditionMonitoring**/submodel/submodelElements/**Status** /**value**	
Headers	Fiware-Service	Test
	Fiware-ServicePath	
Payload	No payload	
Return values		
code	200	Success: Retrieves JSON object including ***value***, and *valueType* for DRobot Status property
	{ "value": 0, "valueType": { "dataObjectType": { "name": "string"} } }	

4.2.2. Asynchronous data query/retrieve

The asynchronous query/retrieve functionality allows the SFM/DSS (or any other external ERP/Data exploitation system) to automatically receive requested information every time any update on it happens and when it happens. This is an implementation of the Publish/Subscribe mechanism described in Section 3.1.2 supported by the CoRoSect Context Broker instance, using the NGSI API. This asynchronous communication is used by CoRoSect Integrated System V1 for:

- Automatically notify the SFM/DSS of any new update on DTs properties, for continuously monitoring the Shop Floor status and commands' flows
- Support the commands' data flows between the context broker (and the registered DTs) and the corresponding connectors (OPC-UA and/or MQTT ones), as well as between the SFM/DSS components and the Context Broker. This is done by triggering automatic notifications when the DT Operation elements are written by the component's server (as a response) or by the SFM/DSS (as a command request).

The following test registers the endpoint (http://CoRoSect_Server/poster/notifications) created to receive POST notifications, creating a subscription to the property "Status" of the DRobot (AASDRobot).

Create and register a Subscription to the DRobot Status value		
POST	[*BaSyx Server*]/subscription	
Headers	Fiware-Service	Test
	Fiware-ServicePath	/test
Payload	{ "description":"Validate the Asynchronous Pub_Sub implementation", "subject":{ "entities":[{ "idPattern":"**Status:AASDRobot**", "type":"I4SubmodelElementProperty" }], "condition":{ "attrs":["**value**"] } }, "notification":{ "http":{"url":"**http://CoRoSect_Server/poster/notifications**"}, "attrs": ["**value**"] } }	
Return values		
code	201	Success (Subscription properly executed)
		Header: subscriptionID: **627d3ad2f9879f2055f87a74**

And this is the notification received when DRobot status changes to “0” value.

NOTIFICATION		
POST	[*http://CoRoSect_Server/poster/notifications*]	
Headers	Fiware-Service	test
	Fiware-ServicePath	/test
Payload	{ "subscriptionId": "**627d3ad2f9879f2055f87a74**", "data": [{ "id": "**urn:ngsi-v2:RAMI40:I4SubmodelElementProperty:AssetConditionMonitoring:Status:AASDRobot**", "type": "I4SubmodelElementProperty", "value": { "type": "Number", **"value": 0**, "metadata": {} } }] }	
Return values		
code	200/201	Success (notification received)

5. Simple orchestration tests

The section describes the set of test scenarios that involve SFM/DSS. These are intended to validate the combined operations between components managed from the SFM/DSS and by using the IMS to invoke corresponding operations. Successfully running workflow scenarios ensures the safe operation of the functionality provided by the shop floor components. Basic tests are the building block for a combined test which is already like a real use case scenario. The following shopfloor components were covered:

- Staking/De-stacking Robot (DRobot)
- Manipulation Robot (MRobot) with Visual Inspection (VI) attached module
- Routes Manager (RM) and indirectly, the Objects Detector (OD)
- AGV

All test scenarios are saved as Business Process Model and Notation (BPMN) files, which can be uploaded by a human operator and executed through the SFM/DSS web application.

5.1. DRobot – Test Scenarios

The first test scenario DRobot-01 (Figure 19), checks if the DRobot receives the correct initial 3d-Dimension (*GridDescription*) of stacked pallet crates or for de-stacking on another pallet later. This is necessary to avoid later problems with the optional looped stacking/de-stacking procedure.

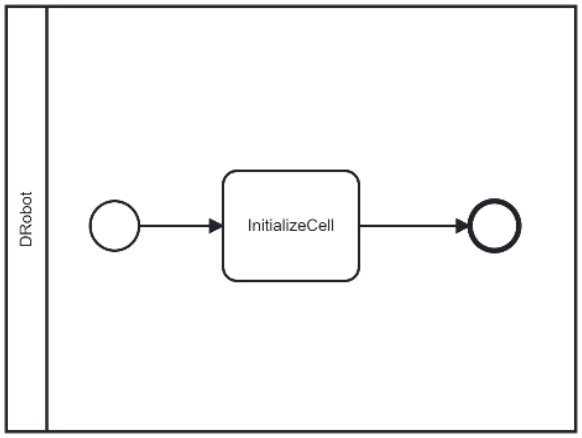

Figure 19. Test Scenario: DRobot-01

The second test scenario DRobot-02 (Figure 20), checks with a more comprehensive workflow the correct execution procedure. The operation *InitilaizeCell* informs DRobot about the current 3d-Dimension (*GridDescription*) for pallets to be stacked/de-stacked. The operation *HomePosition* ensures safe collaboration with the MRobot when they share the workspace. One de-stacking and one stacking operation will be executed to test if the operations can be successfully executed.

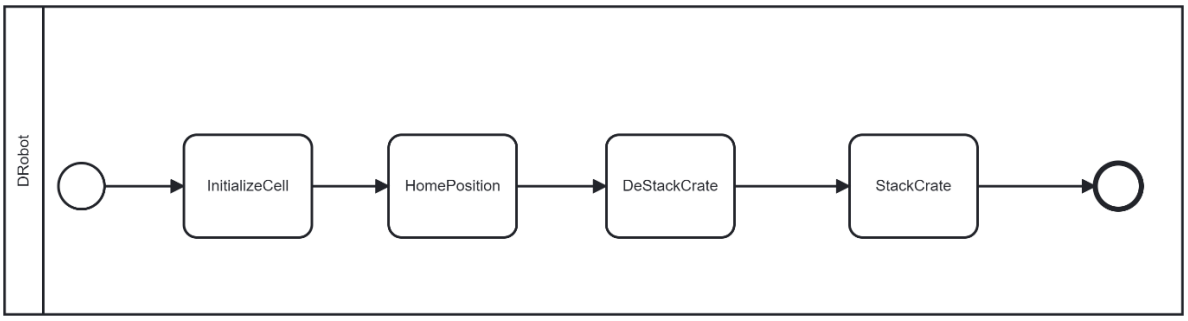

Figure 20. Test Scenario: DRobot-02

The third test scenario DRobot-03 (Figure 21), enhances the previous test scenario DRobot-02 by adding a loop for de-stacking/stacking pallets. The abortion condition “Has more crates” is related to the input variable *GridDescription* of operation *InitializeCell.*

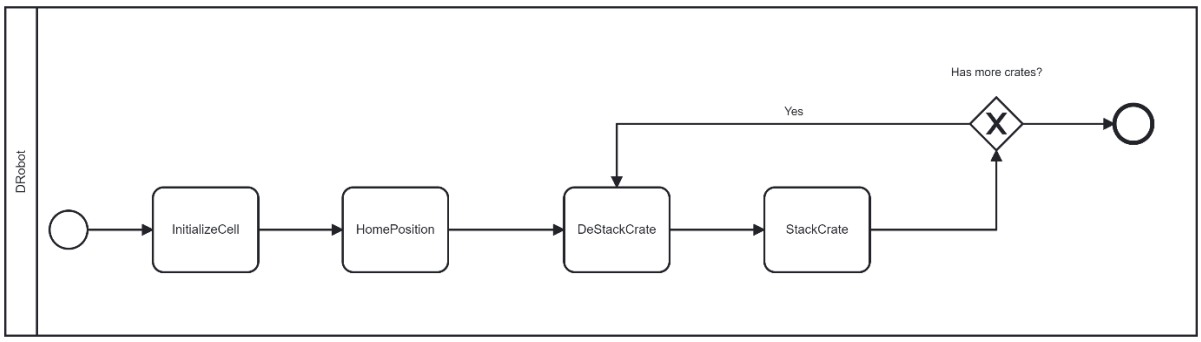

Figure 21. Test Scenario: DRobot-03

5.2. MRobot with VI – Test Scenario

The test scenario MRobot-01 (Figure 22), checks if operations can be successfully executed. The operation *HomePosition* ensures safe collaboration with the DRobot when they share the workspace. The exemplary operation “Take sample” is a combined operation of MRobot with VI (the visual inspection tool).

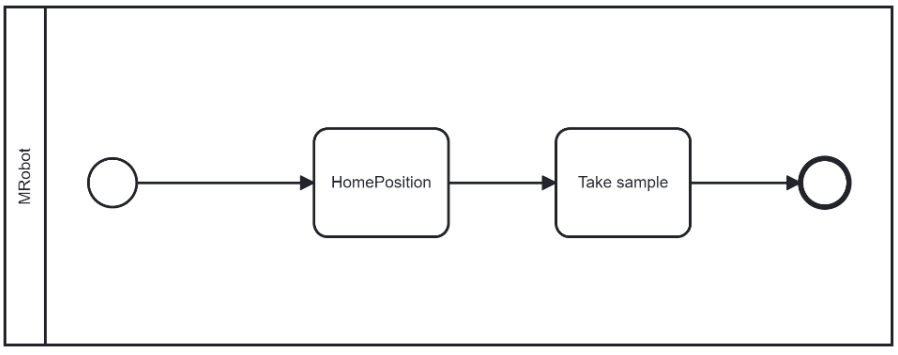

Figure 22. Test Scenario: MRobot-01

5.3. RouteManager – Test Scenarios

The first test scenario RouteManager-01 (Figure 23), checks if a new route can be successfully created and finally started. The operation *StartRoute* normally triggers the AGV to move to its destination.

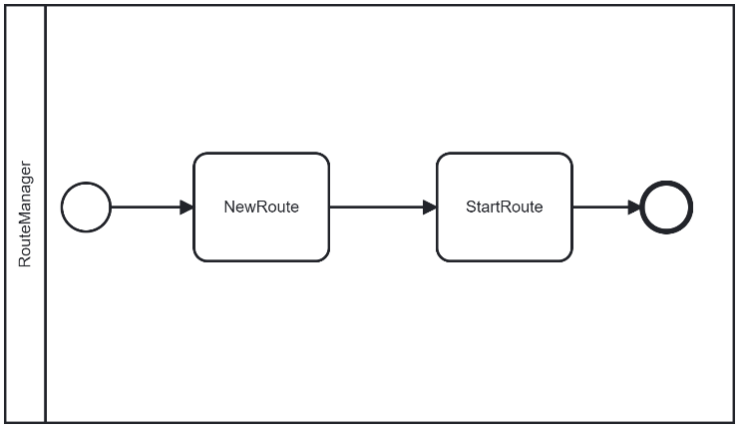

Figure 23. Test Scenario: RouteManager-01

The second test scenario RouteManager-02 (Figure 24), enhances the previous test scenario RouteManager-01 by adding a condition check if the route is valid. Otherwise, it will be retried to create a new route.

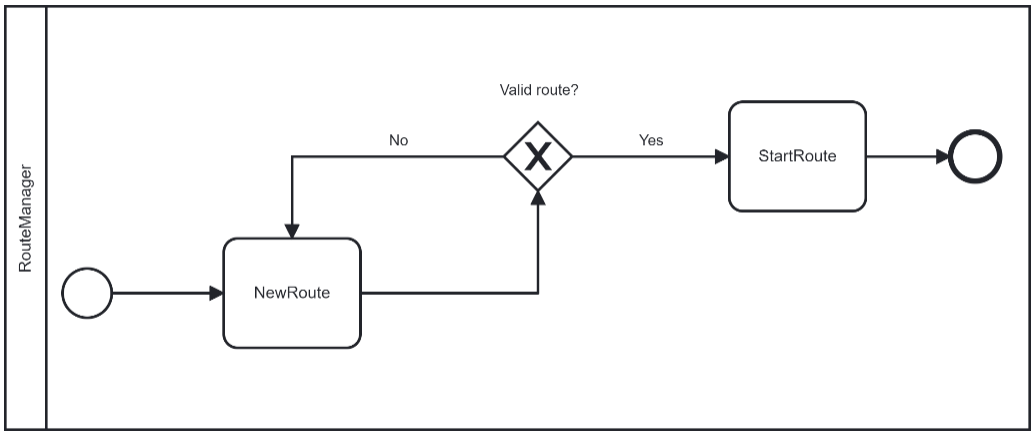

Figure 24. Test Scenario: RouteManager-02

5.4. AGV – Test Scenario

The initial test scenario AGV-01 checks if the AGV can successfully pick and drop pallets by executing the corresponding operations.

5.5. RouteManager and AGV – Test Scenarios

The first test scenario “RouteManager with AGV - 01” (Figure 25), checks if the AGV can pick a pallet, move it from point A to point B, and drop the pallet on point B. A new route was created and started at the *RouteManager* to move the AGV from point A to point B with the pallet.

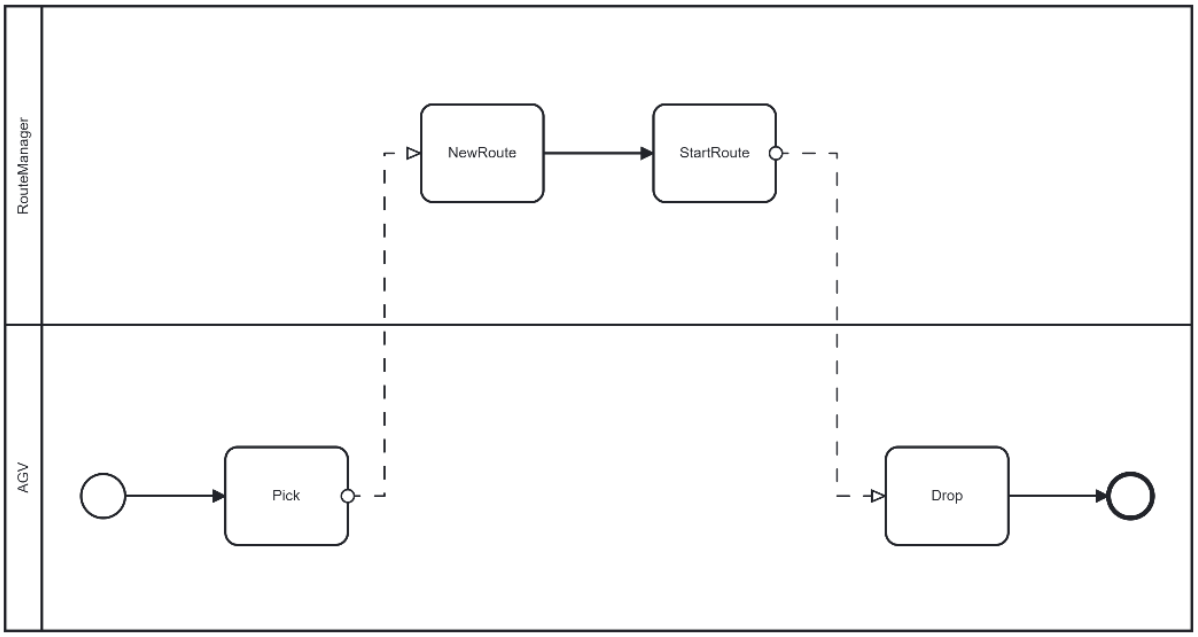

Figure 25. Test Scenario: RouteManager with AGV - 01

The second test scenario "RouteManager with AGV – 02", enhances the previous test scenario "RouteManager with AGV - 01" by adding the condition logic from RouteManager-02. It additionally retries to create a route if not valid through the *RouteManager*.

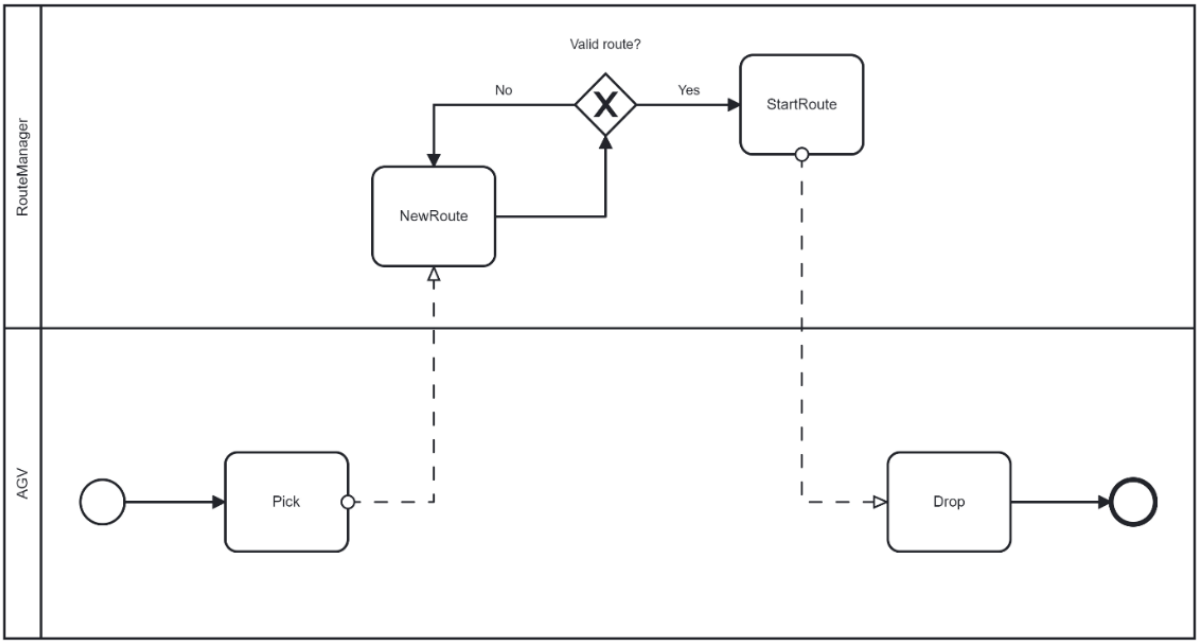

Figure 26. Test Scenario: RouteManager with AGV - 02

5.6. Combined test scenario

The combined test scenario (Figure 27) involves all shop floor components necessary to create later use cases for the pilots. It can be seen as a template. This combined test scenario is based on the previous test scenarios presented. It involves DRobot, MRobot with VI, RouteManager, and AGV.

Table 32 is extracted from the IMS registries and corresponds with the commands' sequence executed during the integration tests carried out within the farm's pilots, where all the physical shop floor

components are present and connected to the same shop floor network. It illustrates the combined test scenario represented in Figure which represents the baseline orchestration between the AGV, the Routes' Manager plus the Objects' detector, the D-Robot and the M-Robot which supports all the Use Cases. This sequence is triggered and managed by the Shop Floor Manager and the Decision Support System, using the IMS interfaces to interact with the real shop floor devices. The *Input Variable*, *Operation Results* and *Output Variable* columns are only represented by a json payload when are provided by the corresponding command input and/or the command output, as the actual content would make the table unreadable.

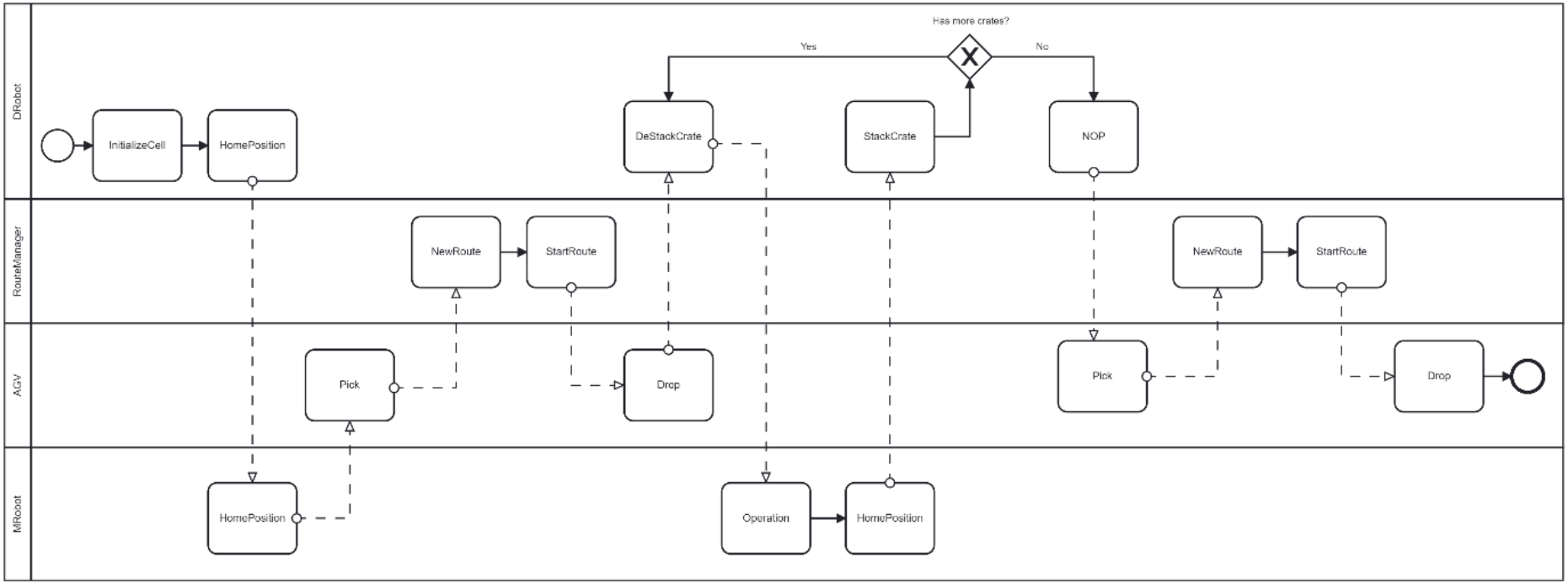

Figure 27. Combined test scenario

Table 32. Commands' sequence for Use Case 1 Shop Floor components' Orchestration. Triggered from the SFM and registered in the IMS.

Input/Output	Time	Request ID (Command ID)	Entity (Submodel:Command:Device_AAS)	Command	Input Variable	Operation Result	Output Variable
Command IN -->	15/06/2023 16:24	d51a5ede-83b0-474e-baff-47b5cdbf5f37	OperationalCapability:pick:**AGV1**	**pick**	*{JSON Input}*		
Response <-- OUT	15/06/2023 16:24	d51a5ede-83b0-474e-baff-47b5cdbf5f37	OperationalCapability:pick:**AGV1**	**pick**			*{JSON Output}*
Command IN + OUT	15/06/2023 16:24	a38ddd28-b0a5-493d-96e0-dbaa37d0474b	OperationalCapability:NewRoute:**RouteManager**	**NewRoute**	*{JSON Input}*	*{JSON Output}*	
Command IN + OUT	15/06/2023 16:26	b0e4253e-6ff4-4516-903f-58a4dfebd0aa	OperationalCapability:StartRoute:**RouteManager**	**StartRoute**	*{JSON Input}*	*{JSON Output}*	
Command IN -->	15/06/2023 16:26	b3ac38f6-6d29-433e-9f98-ae02e4478528	OperationalCapability:drop:**AGV1**	**drop**	*{JSON Input}*		
Response <-- OUT	15/06/2023 16:26	b3ac38f6-6d29-433e-9f98-ae02e4478528	OperationalCapability:drop:**AGV1**	**drop**			*{JSON Output}*
Command IN -->	15/06/2023 16:27	5b874925-ec29-4225-ab92-614781a0aca1	OperationalCapability:ExecuteGeneralTask:**AASDRobot**	**ExecuteGeneralTask**	*{JSON Input}*	*{JSON Output}*	
Response <-- OUT	15/06/2023 16:27	5b874925-ec29-4225-ab92-614781a0aca1	OperationalCapability:ExecuteGeneralTask:**AASDRobot**	**ExecuteGeneralTask**		*{JSON Output}*	*{JSON Output}*
Command IN -->	15/06/2023 16:27	3063e6b4-2f89-4181-aca5-12869e1c7da7	OperationalCapability:DeStackCrate:**AASDRobot**	**DeStackCrate**	*{JSON Input}*	*{JSON Output}*	
Response <-- OUT	15/06/2023 16:27	3063e6b4-2f89-4181-aca5-12869e1c7da7	OperationalCapability:DeStackCrate:**AASDRobot**	**DeStackCrate**		*{JSON Output}*	*{JSON Output}*
Command IN -->	15/06/2023 16:29	e402a402-83e6-45a3-8aac-74bccb0b2cd6	OperationalCapability:MRobotExecuteTask:**AASMRobotVI**	**MRobotExecuteTask**	*{JSON Input}*	*{JSON Output}*	
Response <-- OUT	15/06/2023 16:29	e402a402-83e6-45a3-8aac-74bccb0b2cd6	OperationalCapability:MRobotExecuteTask:**AASMRobotVI**	**MRobotExecuteTask**		*{JSON Output}*	*{JSON Output}*
Command IN -->	15/06/2023 16:33	c6fcfe2e-a09c-4167-88f6-54a1053814eb	OperationalCapability:StackCrate:**AASDRobot**	**StackCrate**	*{JSON Input}*	*{JSON Output}*	
Response <-- OUT	15/06/2023 16:33	c6fcfe2e-a09c-4167-88f6-54a1053814eb	OperationalCapability:StackCrate:**AASDRobot**	**StackCrate**		*{JSON Output}*	*{JSON Output}*
Command IN -->	15/06/2023 16:35	e7b651c7-cee4-4626-b4d5-632faeead0d2	OperationalCapability:pick:**AGV1**	**pick**	*{JSON Input}*		
Response <-- OUT	15/06/2023 16:35	e7b651c7-cee4-4626-b4d5-632faeead0d2	OperationalCapability:pick:**AGV1**	**pick**			*{JSON Output}*
Command IN + OUT	15/06/2023 16:36	aedc4199-2cdf-415c-bd7d-4ab4e6becbe8	OperationalCapability:NewRoute:**RouteManager**	**NewRoute**	*{JSON Input}*	*{JSON Output}*	
Command IN + OUT	15/06/2023 16:36	e27de63c-2395-411a-bd4d-2e7d79230e30	OperationalCapability:StartRoute:**RouteManager**	**StartRoute**	*{JSON Input}*	*{JSON Output}*	
Command IN -->	15/06/2023 16:38	2714a9f9-a8a8-4e34-bbf4-f5012273accc	OperationalCapability:drop:**AGV1**	**drop**	*{JSON Input}*		
Response <-- OUT	15/06/2023 16:38	2714a9f9-a8a8-4e34-bbf4-f5012273accc	OperationalCapability:drop:**AGV1**	**drop**			*{JSON Output}*

6. Conclusions

The set of results shown within this document represents the summary and the final outcomes of a longer process carried out with each of the CoRoSect system components, including here physical robot devices, deployed servers, developed software connectors, interfaces' implementations, and IMS and SFM/DSS building blocks. This integration process involved all CoRoSect technical staff (Shop Floor components' providers and MES developers) and many failed tests checking communication protocols, connectors' configurations, programming bugs and a lot of concepts' discussions till we get a final success reading each required property and running each implemented command according to the RAMI4.0 specifications.

In a first stage, this process focused on each single component integration with the IMS. In this way, the OPC and MQTT connection; each shop floor component's Digital Twin plus their corresponding cell controller and server; the data gathering, storage and catering services and the commands' flows were successfully validated. For this stage, the cloud instance of the CoRoSect Integrated system was used, and so, also validated.

For a second stage, the orchestration of the real shop floor, this is: the basic sequence of related tasks executed by each corresponding robot and system, everything commanded and controlled by the CoRoSect's Shop Floor Manager; was also partially validated. For this second stage, all robots and systems involved in the use cases were needed to be present and connected to the same network, so this orchestrations' tests were only possible during the farms' pilots. In this sense, two independent instances of the CoRoSect's Integrated systems were deployed on each farm to evaluate the installation of the required software pieces. For these reasons, current document was delayed till the second pilot, so corresponding tests were done. Within these tests, most of the technical integration and orchestration issues were successfully polished and finally validated. Nevertheless, there were some orchestration concepts related to the tasks flows progress' tracking and concrete execution of commands related to the AGVs routing, that prevent the achievement of 100% of the full orchestration. These issues have been properly identified and will be solved, tested, and validated during the next two final pilots.

From the full validation process executed so far, we mainly conclude:

- Although data and commands flows were successfully validated, the overall Information exchange is in general, inefficient. Too much static information is moved within each message. This approach is valid for the pilots' framework, with very little devices integrated/connected but the unnecessary payloads will increase when adding wider farms. Connectors should be modified, and storage strategy should be revisited to move and store only changing values, while keeping RAMI4.0 compliance.
- The orchestration process must be polished and proper tracking properties must be identified and standardised among the defined DTs. Current implementation allows each provider to name and use their own properties that monitors their tasks' progress. This makes the Shop Floor Manager to program customised flows' controls for each integrated component. Standard task monitoring properties will result in simpler, more efficient control mechanisms for the SFM.

These two concluding issues will be considered for next full system version inf D9.3 and its validation in D9.5, as well as for future evolutions of the CoRoSect System, beyond CoRoSect project.

After this first full validation process, the CoRoSect System V1 is ready for Use Cases performance and so, evolve to its final version.

CoRoSect

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