IntellioT

Deliverable D2.6
High level architecture (final version)

Deliverable release date	11/10/2022
Authors	1. SIEMENS: Andreas Zirkler, Andreas Ziller, Arne Bröring, Vivek Kulkarni 2. EURECOM: Jérôme Härri 3. AAU: Beatriz Soret, Lam Nguyen 4. UOULU: Sumudu Samarakoon 5. TTC: Martijn Rooker 6. TSI: Andreas Brokalakis, Vassilis Amourgianos, Babis Savvakos 7. PHILIPS: Nancy Irisarri Mendez, Anca Bucur 8. SANL: Konstantinos Fysarakis, Ioannis Vezakis 9. HSG: Simon Mayer, Jérémy Lemée 10. HOLO: Carina Pamminger 11. AVL: Holger Burkhardt, Wolfgang Hollerweger 12. MYW: Giancarlo Bo, Alessandro Sagoleo 13. IKH: Nikos Frangakis 14. VID: Alexis Fourlis 15. TRI: Federico Viani
Editor	Konstantinos Fysarakis (SANL)
Reviewer	Arne Bröring (SIEMENS), Martijn Rooker (TTC)
Approved by	PTC Members: (Vivek Kulkarni, Konstantinos Fysarakis, Sumudu Samarakoon, Beatriz Soret, Arne Bröring, Maren Lesche) PCC Members: (Vivek Kulkarni, Jérôme Härri, Beatriz Soret, Mehdi Bennis, Martijn Rooker, Sotiris Ioannidis, Anca Bucur, Georgios Spanoudakis, Simon Mayer, Filippo Leddi, Holger Burkhardt, Maren Lesche, Georgios Kochiadakis)
Status of the Document	Final
Version	1.0
Dissemination level	Public

Table of Contents

ACRONYMS AND DEFINITIONS

Acronym	Definition
5G NR	5th Generation New Radio
AODV	Ad-hoc On-demand Distance Vector
AMPQ	Advanced Message Queuing Protocol https://www.amqp.org/
API	Application Programming Interface
AI	Artificial Intelligence
AR	Augmented Reality
AAA	Authentication, Authorization and Accounting
AAA	Authentication, Authorization, and Accounting
CAN	Controller Area Network
CN	Core Network
CPET	Cardiopulmonary exercise test
D2D	Device to Device
DLT	Distributed Ledger Technology
EMS	Edge Management System
ECG	Electrocardiogram
eMBB	enhanced Mobile Broadband
ETH	Ether crypto currency
ETSI	European Telecommunications Standards Institute
FL	Federated Learning
GIS	Geographic Information System
GPS	Global Positioning System
gNB	gNodeB
GUI	Graphical User Interface
LLDP	Link Layer Discovery Protocol
HIL	Human in the Loop
HW	Hardware
Hypermedia MAS	Hypermedia Multi-agent System
ID	Identification
IMU	Inertial Measurement Unit
ICT	Information and Communication Technology
IT/OT	Information Technology / Operation Technology
IAKM	Infrastructure Assisted Knowledge Management
IO	Input Output
IDE	Integrated Development Environment
IPR	Intellectual Property Right
ISAR	Interactive Streaming for Augmented Reality
IMSI	Internal Mobile Subscriber Identity
IoT	Internet of Things
IPSec	Internet Protocol Security
IPFS	InterPlanetary File System
IDS	Intrusion Detection System
JSON	JavaScript Object Notation
JSON-LD	JSON for Linking data; JSON-LD - JSON for Linking Data
JWT	JSON Web Token; JSON Web Tokens - jwt.io
KPI	Key Performance Indicator
LoS	Level-of-Service
LBS	Location-based Services
LL-MEC	Low Latency Multi-access Edge Computing

ML	Machine Learning
MAS	Multi-Agent System
MES	Manufacturing Execution System
MEC	Mobile Edge Computing
mMTC	massive Machine-Type Communication
MTD	Moving Target Defences
NB-IoT	Narrow Band IoT
NFV	Network Function Virtualization
TUN	network TUNnel (Internet protocol/ Link Layer)
NN	Neural Network
NG IoT	Next Generation Internet of Things
OLSR	Optimized Link State Routing
OEM	Original Equipment Manufacturer
QoS	Quality of Service
RAN	Radio Access Network
RNTI	Radio Network Temporary Identifier
RRM	Radio Resource Management
RFID	Radio-Frequency Identification
RTLS	Real-Time Locating System
RL	Reinforcement Learning
RPM	Remote Patient Monitoring
RPC	Remote Procedure Call
ROS	Robot Operating System: https://www.ros.org
SAP	Security Assurance Platform
TD	Things Description of W3C WoT
TSN	Time-Sensitive Networking
TLS	Transport Layer Security
UR5	Robot arm https://www.universal-robots.com/products/ur5-robot
UC	IntellIoT Use Case (1: agriculture, 2: healthcare, 3: manufacturing)
URLLC	Ultra Reliable Low Latency Communication
UML	Unified Modelling Language
UDP	User Datagram Protocol
UE	User Equipment
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
VR	Virtual Reality
WoT	Web of Things
WP	Work Package of IntellIoT project
W3C	World Wide Web Consortium
W3C WoT TD	World Wide Web Consortium Web of Things Things Description

1 INTRODUCTION

This deliverable, being the final output of Task 2.3 ("Architecture specification & interoperability"), aims to specify the final version of the architecture of the IntellIoT framework, the components comprising it, and the key functionalities and interfaces of these components, updating the content presented in the first version of this deliverable, i.e., D2.3 - "High level architecture (first version)".

These efforts are aligned with the project's 5th Objective pertaining to the development of a reference implementation of the IntellIoT framework, consisting of (a) mechanisms for the human-in-the-loop and assuring trust, (b) a novel architecture (i.e., a Hypermedia MAS) for self-aware and (semi-)autonomous IoT applications and (c) a dynamic IoT/edge infrastructure in closed-loop with heterogeneous networks.

As such, and since the delivery of the architecture sets the foundations of the IntellIoT framework and the pertinent implementation and validation efforts, there is a strong interplay between the work within Task 2.3 (and the content of D2.6 herein, by extension) and the majority of other Work Packages (WPs) and Tasks of the project. In more detail, Task 2.3 receives valuable input from Task 2.1 ("Use cases specification & Open Call definition") and Task 2.2 ("Technology analysis & requirements specification") concerning the intricacies of the project's use cases, the functional and non-functional requirements, and other technological and demonstration - specific aspects that help update and refine the rough architecture defined during the proposal phase. Once specified, the final Architecture will provide updated input to inform the implementation tasks within WP3 ("Distributed, self-aware IoT applications & human-defined autonomy") and WP4 ("Efficient, reliable and trustworthy computation & communication infrastructure"), with specifics about the component capabilities, interfacing requirements, deployment/infrastructure intricacies and other necessary technical characteristics for the 2nd and final release of these components, within Cycle 2 of the project. Furthermore, as the content presented herein concerns IntellIoT's Cycle 2, the results and feedback from the Cycle 1 demonstration and evaluation carried out within the three Tasks of WP5 ("System integration & evaluation") provided vital input to Task 2.3 to allow the refinement, improvement, and enrichment of the framework's reference Architecture. These relationships are visualised in the PERT chart shown in Figure 1, whereby T2.3 and its linked tasks, as described above, are highlighted.

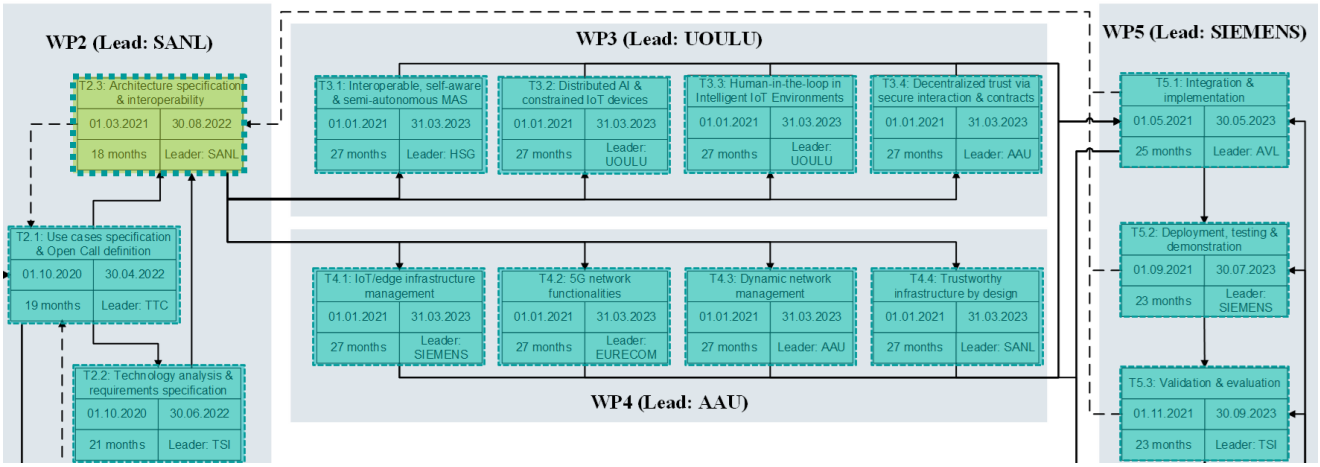

Figure 1. IntellIoT's PERT chart, with highlights on Task 2.3's role and linked tasks.

In addition to the detailed specification of IntellIoT's architecture, this deliverable: (1) identifies the framework's technical requirements (building upon work in previous WP2 tasks and the architecture specification efforts); (2) highlights how the defined architecture and its building blocks provide coverage in terms of the project's objectives and requirements, and (3) studies the alignment of IntellIoT's architecture with other established IoT initiatives and their architectures.

To present the above, the deliverable is organized as follows:

- Section 1 (this section) provides the introductory remarks along with a description of the process followed for the architecture definition and some background information on the chosen architectural view model.

- Section 2 presents the IntellIoT Architecture in detail, covering the key pillars of the project, along with different views that allow a holistic representation of the framework, its components and its functionality both in an application agnostic as well in a use-case specific manner.
- Section 3 presents the frameworks Technical Requirements, considering the requirements defined within Task 2.2 but also the new technical needs (e.g., component specifications, planned features and interactions) identified during the architecture definition efforts.
- Section 4 provides a mapping of the different components comprising IntellIoT to the project's Objectives and the different types of Requirements. This allows for checking that the defined components and their capabilities are designed to provide a full coverage in terms of satisfying said Objectives and Requirements (i.e., there are no design-time omissions).
- Section 5 highlights key initiatives in the IoT Architectures' and Edge landscape, studying the alignment and mapping of the IntellIoT architecture to the identified initiatives.
- Finally, Section 6 provides the concluding remarks and pointers to the next steps.

Before presenting the architecture, updates compared to the previous deliverable, and some background information on the process and chosen architecture definition model are presented in the subsections that follow.

1.1 Modifications compared to deliverable D2.3

Considering the delta to the previous version of the deliverable, i.e., D2.3 - "High level architecture (first version)", the new *content presented within this final Task 2.3 deliverable include:*

- The Introduction section also features a new section (1.1) presenting the overall WP2 process and Task 2.3's positioning. Other minor updates were also made to the introduction & conclusions' sections, to bring them up-to-date with the current deliverable, as the last output of Task 2.3.
- A major update of the IntellIoT Architecture presented in Section 2, to derive the final IntellIoT architecture, which now encompasses:
    - updates to the various components and their integration, based on feedback from Cycle 1,
    - final Use Cases' specification, as updated and documented in D2.4 – "Use case specification & Open Call definition (final version)"
    - final set of Requirements, as updated and documented in D2.5 - "Technology analysis & requirements specification (final version)"
    - changes & updates introduced through component development and integration efforts (e.g., changes to implementation approaches, technologies used)
    - the integration of the technologies that the Open Call 1 winners brought to IntellIoT.
- An update to the Technical Requirements defined in Section 3 to align them with the final architecture and its building blocks (as presented in Section 2), and the updates to the overall requirements defined in D2.4.
- An update to Section 4, with new mappings to project objectives & requirements, covering the updated set of components (defined in Section 2) and the updated set of requirements (defined in D2.4 & in Section 3).
- Other minor updates were also made to the introduction & conclusions' sections, to bring them up-to-date with the current deliverable, as the last output of Task 2.3.

1.2 Architecture specification within the overall Work Package 2 Process

The overall process followed within Work Package 2, is presented in Figure 2. Starting with the proposal-phase material and guidelines (Use Case specifications, the adoption of a Design Thinking methodology, the identified partner technologies & the early architecture vision), the consortium, in the context of Task 2.1 ("Use cases specification & Open Call definition"), refined the use cases, deriving Collaborative IoT, Human-in-the-loop and Trustworthiness scenarios (i.e., 3 scenarios per Use Case), and associated key scenes. These were analysed to

define the Open Calls' positioning, but also (within Task 2.2 – "Technology analysis & requirements specification") to drive the end user workshops that provided additional feedback on the Use Cases, allowing to eventually identify the full set of IntellIoT functional, non-functional & technical requirements.

The results of these efforts, carried out in Cycle 1 and revised in Cycle 2 of the project, have all been documented in the relevant deliverables, namely D2.1 – "Use case specification & Open Call definition (first version)", updated in D2.4 – "Use case specification & Open Call definition (final version)", and D2.2 – "Technology Analysis & Requirements Specification (first version)", which was updated in D2.5 – "Technology Analysis & Requirements Specification (final version)". More details on the interactions with end users, in specific, are provided in D6.2 – "Dissemination & ecosystem building (first version)". Finally, Task 2.3, with its final outputs presented herein, closely followed the above activities (even before the formal start of the Task 2.3), ensuring that the architecture definition process (detailed in subsection 1.3 below), encompasses all relevant inputs (e.g., derived requirements) both at the architecture but also at the component specification level.

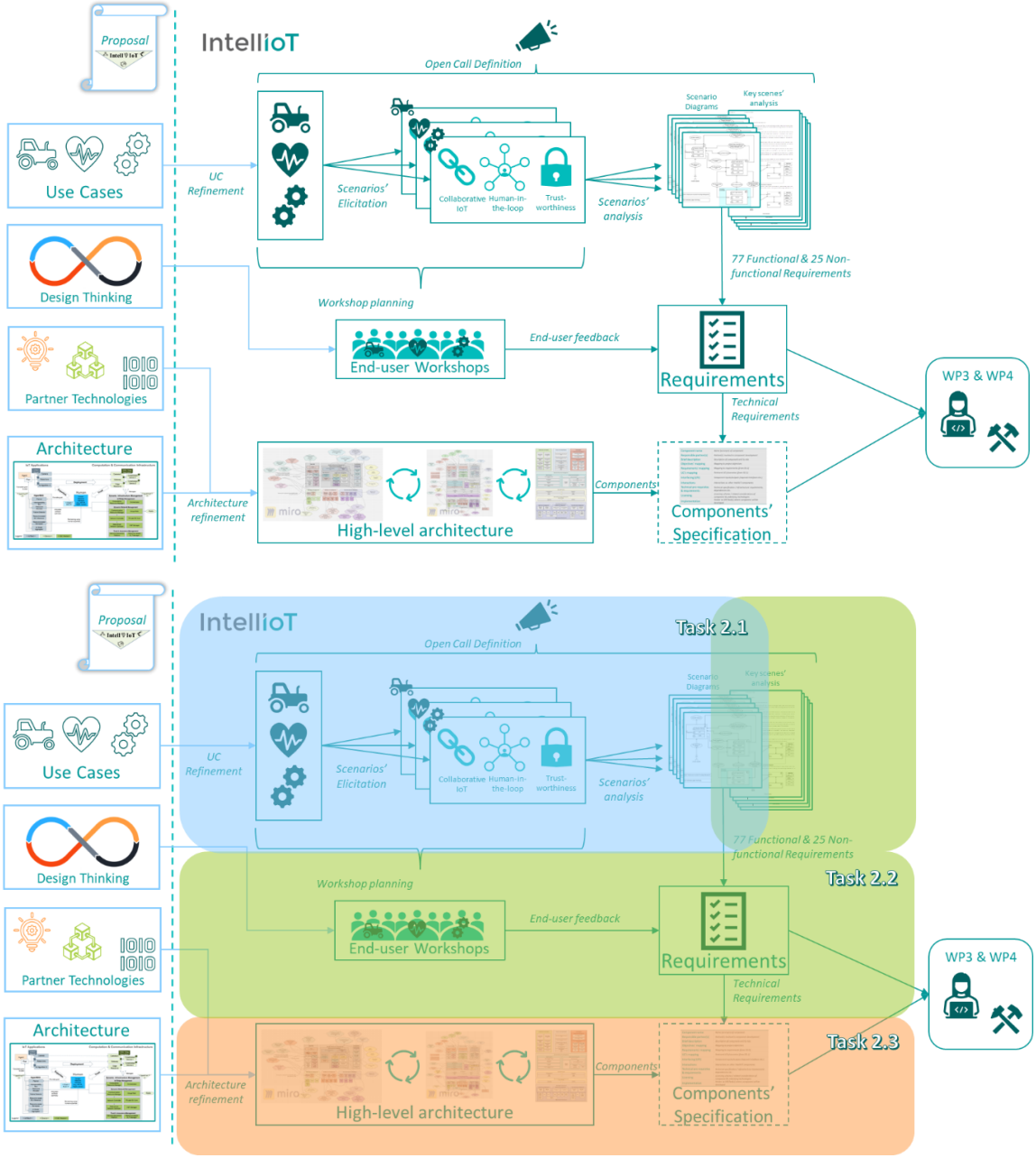

Figure 2. Overall WP2 process (top) & breakdown of activities into individual tasks (bottom)

1.3 Architecture definition process

A six-stage process had been followed to define the IntellIoT Cycle 1 architecture. The steps in this process are shown in Figure 3.

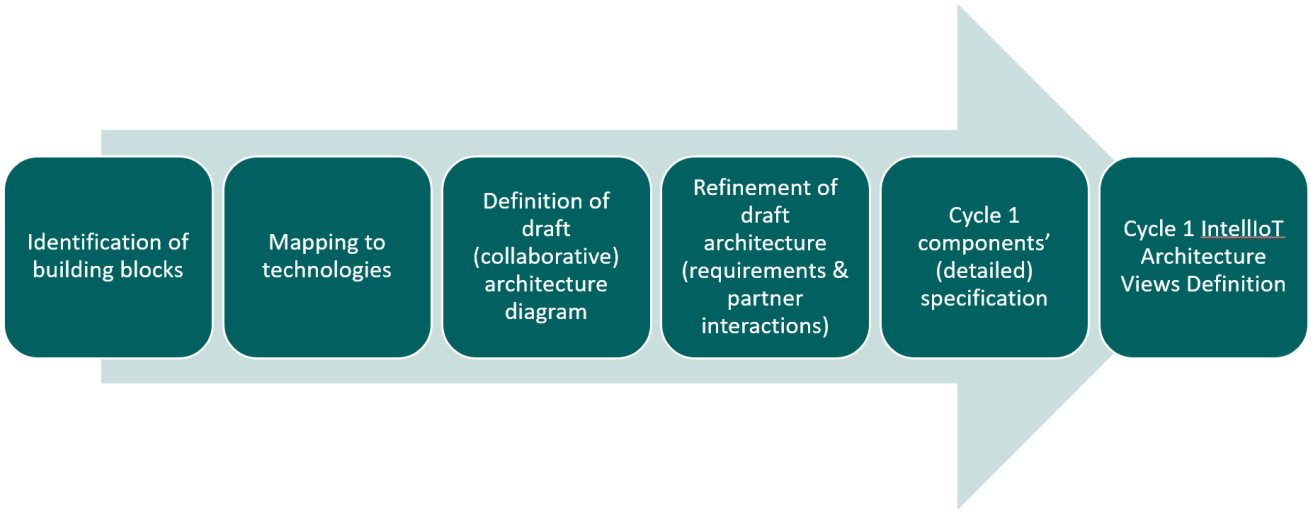

Figure 3. IntellIoT's Cycle 1 Architecture definition process.

The first two steps, namely the identification of the core building blocks and their mapping to the technologies that the consortium partners bring to the project, relied mostly on proposal-phase material (such as the proposal-phase high level architecture shown in Figure 4).

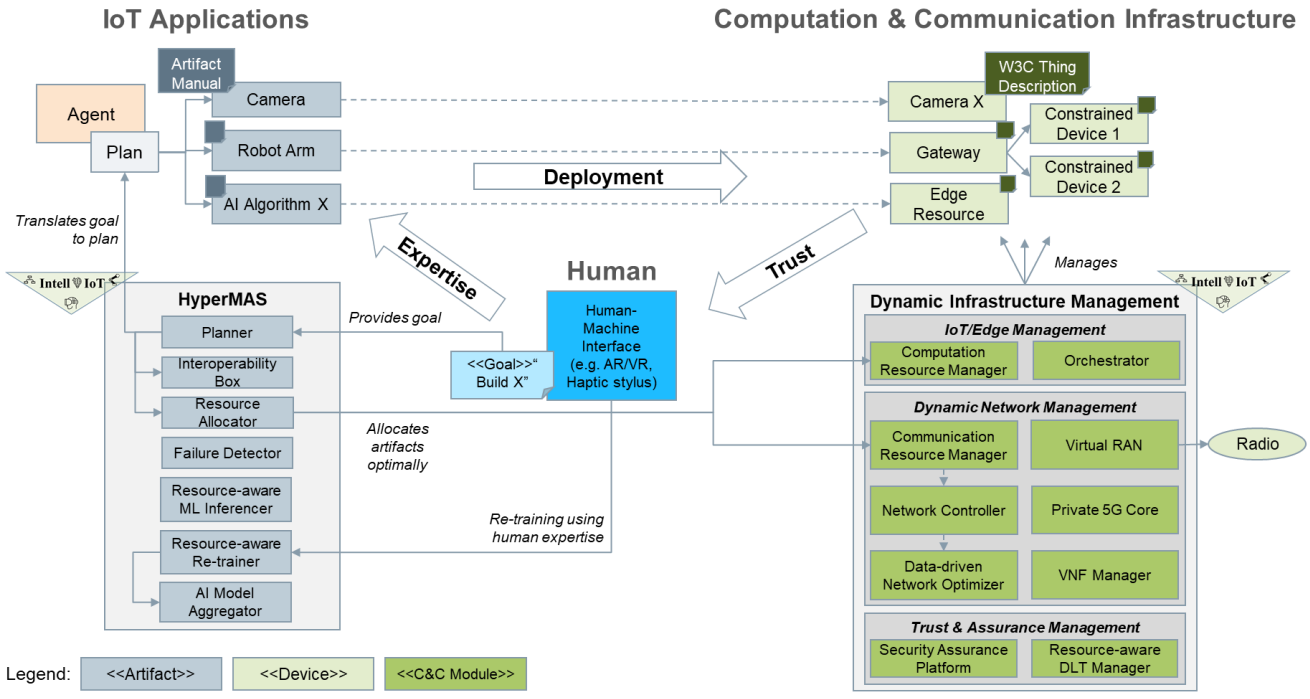

Figure 4. IntellIoT's proposal-phase high-level architecture.

Once these two steps were accomplished (revising them, also considering the input from the work being carried out within Task 2.1 and Task 2.2), an updated draft architecture was produced. Then, the consortium collaboratively worked on refining and enriching this updated draft architecture. To facilitate this interaction and

collaboration-intensive process, the Miro[1] collaborative design platform was employed, allowing partners to work in real time on the same figure. A snapshot of the collaborative Miro board used to define this draft architecture is shown in Figure 5, where bubbles mapping technologies to components as well as various comments to be clarified within the consortium can be seen as well.

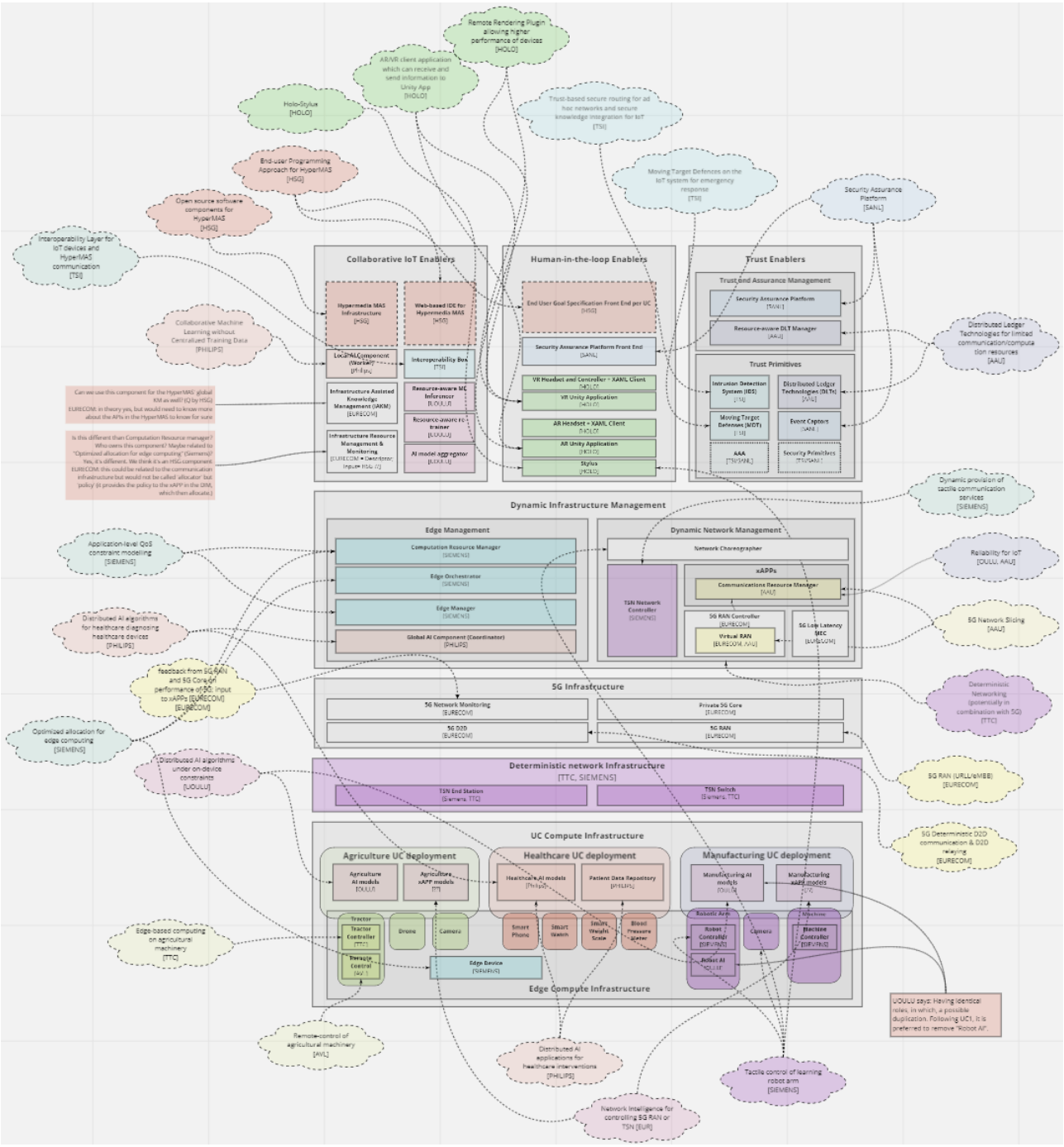

Figure 5. Snapshot of IntelIIoT's draft architecture, with key building blocks and their mapping to technologies, as collaboratively produced by partners on the Miro platform.

[1] https://miro.com/

Once this draft architecture reached a stable state, the fifth step in the process was initiated, whereby individual building blocks had to specified in detail, documenting aspects such as key functions, interfacing, and infrastructure requirements, licensing intricacies, and mapping to project objectives. To streamline this process, a "Component Specification Template" was defined and used by all partners, with such a table being populated for each and every one of IntellIoT's components; this table can be seen in Figure 6 (in italics a description of each field).

Component name	*Name (and acronym) of component*
Responsible partner(s)	*Partner(s) involved in component development*
Brief description	*Description of component and its role*
Objectives' mapping	*Mapping to project objectives*
Requirements' mapping	*Mapping to requirements (from D2.2)*
UC's mapping	*Pertinent UC's/scenarios (from D2.1)*
Interfacing (I/O)	*Component inputs/outputs (exposed interfaces etc.)*
Interactions	*Interactions w. other IntellIoT components*
User interfacing & visualisation	*Any user interfacing / visualisation needs, and if these are inherently provided by component / responsible partner.*
Technical pre-requisites & requirements	*Technical specification / infrastructure requirements, dependencies etc.*
Licensing	*Licensing scheme / related considerations of component & underlying technologies*
Implementation	*Pointer to WP/Task(s) where component will be developed*
Table Version	*V0.1 to V2.0 [V1.0 = Final for R1 of IntellIoT, V2.0 = Final for R2 of IntellIoT]*

Figure 6. IntellIoT's Component Specification Template

The component specification tables of all components, once ready, along with the collaborative draft architecture defined in the previous step, were used as inputs for the last step in the process: defining the architecture in a more structured and formal manner.

To derive the final version of the architecture, as presented herein, the 3 final steps of the above approach have been repeated. Therefore, while the consortium will not go "back to the drawing board", to define the architecture from the beginning, a significant update took place both at the component as well as the overall architecture (i.e., components' integration) level.

Similarly, to Cycle 1, for Cycle 2 the "4+1" architectural view model was used by the consortium (details on this model are provided in the subsection that follows) to define the architecture, therefore updating all 4+1 views to provide the new, final version of it.

1.4 The 4+1 Architectural View Model

When designing a system, multiple "views" must be taken into account. Consider the example of building a house, which typically has floor plans, electrical plans, plumbing plans, etc. All these different views are necessary to properly describe its architecture, as a single architectural style is not sufficient to provide an accurate and detailed enough view that would allow documenting all the details needed to, ultimately, build the house as intended.

Motivated by this, the "4+1 View Model of Architecture" (presented by P. B. Kruchten in his seminal work [1]), was developed to describe a software architecture using five concurrent views. Each of these views addresses a specific set of concerns and describes the system in the viewpoint of different stakeholders, such as end-users, developers, and project managers.

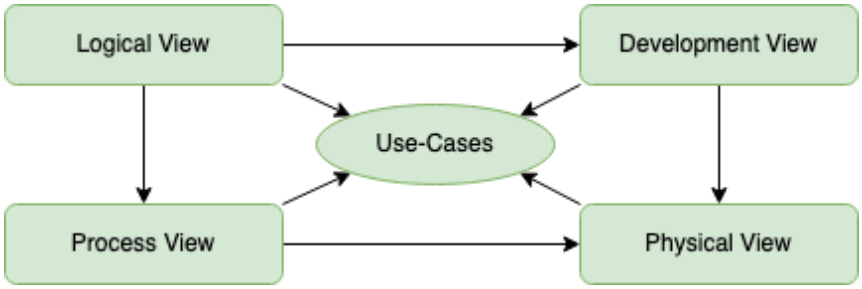

Figure 7. Different views within the 4+1 architectural view model and their relationship

As shown in Figure 7, the views included are: Logical view, Process view, Development view, Physical view and, finally, a Use-Cases view (referred to as the "+1" view). In the subsections below, more details will be provided for each of these views, along with potential examples on how to specify them (as the model is generic, and not restricted to any notation, tool or design method).

1.4.1 LOGICAL VIEW

This view answers **how the system is structured**, and primarily supports the functional requirements. It provides a high-level overview of the system and the components it is comprised of, with directional links providing a sense of how the system is connected and the information flow. UML diagrams typically used to represent this view include class, object, state, or composite structure diagrams (e.g., see Figure 8). In this deliverable, a component diagram was used to represent the logical view, with directional links between the components depicting their interactions and information flow.

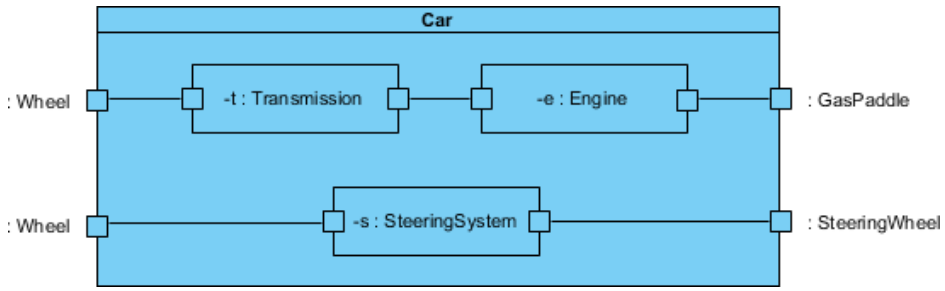

Figure 8. Example of a logical view diagram.

1.4.2 PROCESS VIEW

This view answers **how the system behaves**, addressing concurrency and synchronization aspects. The components depicted here are the same as the software components in the development view, but this time placed in a sequence diagram. Usually, it is represented by sequence, communication, and activity UML diagrams (e.g., see Figure 9). For the IntellIoT Architecture, UML sequence diagrams were selected as the most appropriate form.

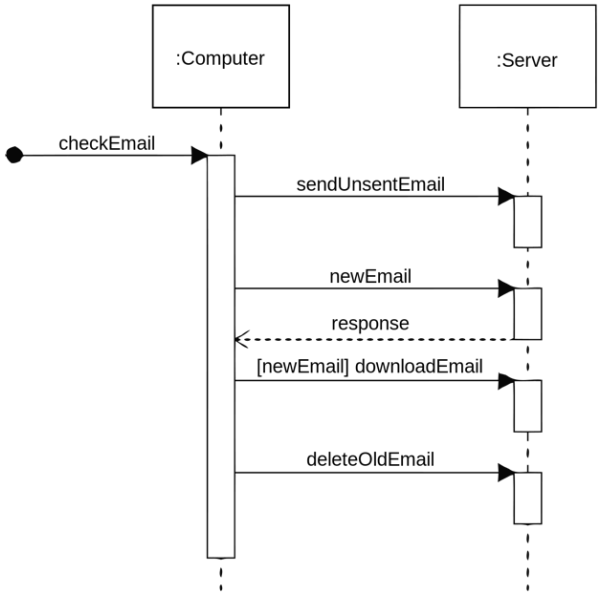

Figure 9. Example of a UML sequence diagram.

1.4.3 DEVELOPMENT VIEW

This view answers **how the system is built**. It focuses on the software component organization and their interfaces. Unlike the logical view, the components depicted here are solely software components. Typically, UML diagrams to represent this view usually include component and package diagrams (e.g., see Figure 10). Herein, all of IntellIoT's software components are depicted in a component diagram, with lollipop connectors linking them to show their interfaces.

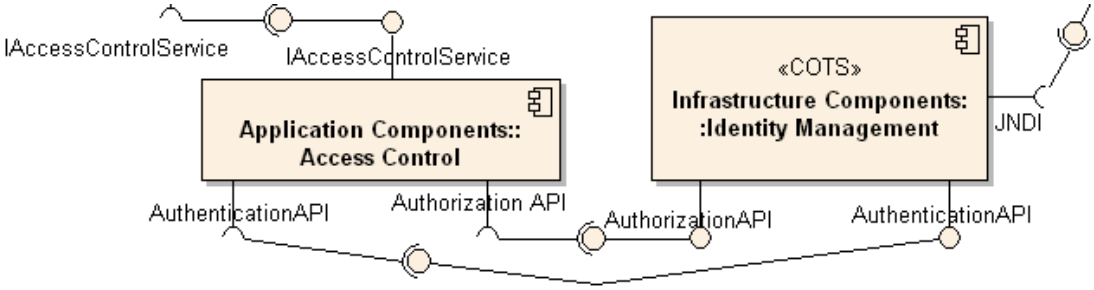

Figure 10. Example of a development view UML component diagram.

1.4.4 PHYSICAL VIEW

This view describes **how the system is deployed** (thus, sometimes also referred to as the "deployment view"), including the hardware and the devices involved in the system. It depicts the mapping of the software components onto their respective processing nodes (servers, virtual machines, containers, edge devices, etc). A deployment diagram can be used to demonstrate this view (e.g., see Figure 11).

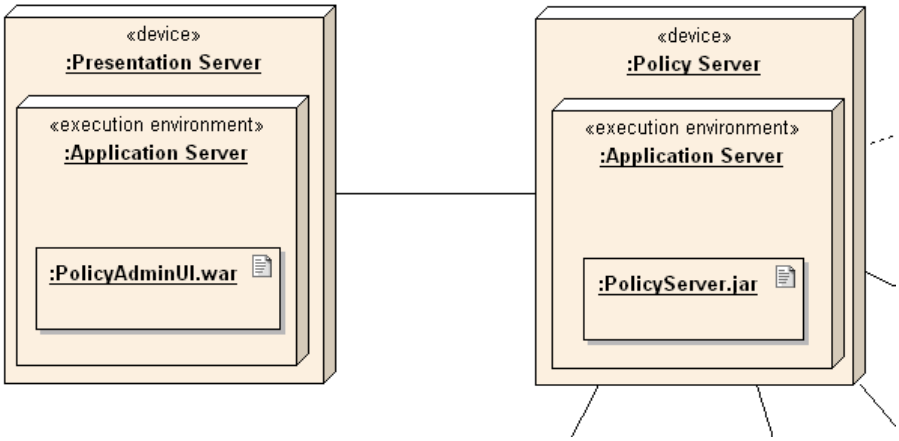

Figure 11. Example of a physical view diagram.

1.4.5 USE-CASES (+1) VIEW

This is the +1 in the "4+1" view model. This view intends to illustrate how the system (architecture) is applied in different use cases (also referred to as scenarios, and "Scenario view"). Usually, these are depicted in forms like object scenario diagrams, object interaction diagrams or UML use case diagrams (e.g., see Figure 12). Such use case diagrams have already been defined for the project's use cases in deliverable D2.1 (updated in D2.4) and further elaborated upon within D2.2 (updated in D2.5). Thus, the +1 view is used to depict the variations, peculiarities, and specific devices of each Use-Case demonstrated using the IntellIoT framework.

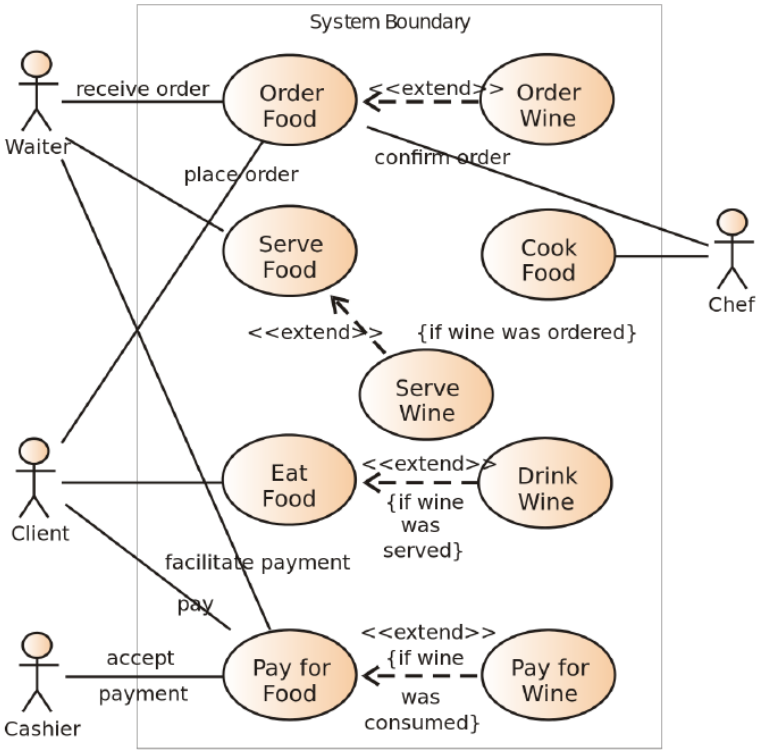

Figure 12. Example of a UML use case diagram

2 THE INTELLIOT ARCHITECTURE

Through the process detailed within subsection 1.3, a high-level view of IntellIoT's logical architecture was derived, as shown in Figure 13. An important consortium decision was to have **the three pillars of the project (i.e., Collaborative IoT, Human-in-the-loop and Trustworthiness) prominently featured in the architecture**, as they are key to the IntellIoT concept.

In total, five core component groups have been identified, with individual components falling into one of the following groups:

(1) **Collaborative IoT enablers**: This group contains the components that realize IntellIoT's Collaborative IoT pillar, focusing on the cooperation of various semi-autonomous entities (tractors, robots, healthcare devices, etc.) to execute multiple IoT applications.
(2) **Human-in-the-Loop enablers**: This group contains the components involved in IntellIoT's Human-in-the-Loop (HIL) pillar, which focuses on involving the human in the process, when necessary, in order to solve complex situations that the system does not yet know how to handle.
(3) **Trust enablers**: This group involves components that are part of IntellIoT's Trust pillar. This pillar focuses on privacy, security, and ultimately building trust into the IntellIoT framework.
(4) **Infrastructure management**: This group is comprised of the computation and communication infrastructure and its management capabilities, which enable the deployment and management of edge applications.
(5) **Use-Case deployment**: This group involves all components which are Use-Case specific, (i.e., pertaining to the use case environment deployment), such as edge devices and their hardware, edge apps, and edge AI models.

It should be noted that, except for the latter group (i.e., group 5, which is there to document and support use-case specific enablers), all other groups (i.e., groups 1 to 4) **feature use-case agnostic enablers that comprise the core IntellIoT framework** and which are potentially usable in all three studied as well as additional NG-IoT use cases.

Nevertheless, as highlighted in subsection 1.4, the architecture of a complex framework such as IntellIoT cannot be visually represented in a single view (such as the one shown in Figure 13), in a manner that is detailed enough to drive implementation and integration, but also maintains its readability. Thus, to holistically present the framework's Architecture, the included components and their characteristics and functions, the 4+1 architectural view model was selected. Furthermore, the four core views (i.e., excluding the "+1" use case view) have been broken down into the corresponding component groups mentioned above: Collaborative IoT, HIL, Trust, and Infrastructure Management. Each of these groups is therefore described separately in the following subsections, with the last view, the Use-Cases, tying them together again in a single logical view.

Finally, Figure 14 shows the same logical view, albeit this time enriched with the core framework enablers contributed by Open Call 1 winner for the framework category, MYW.AI.

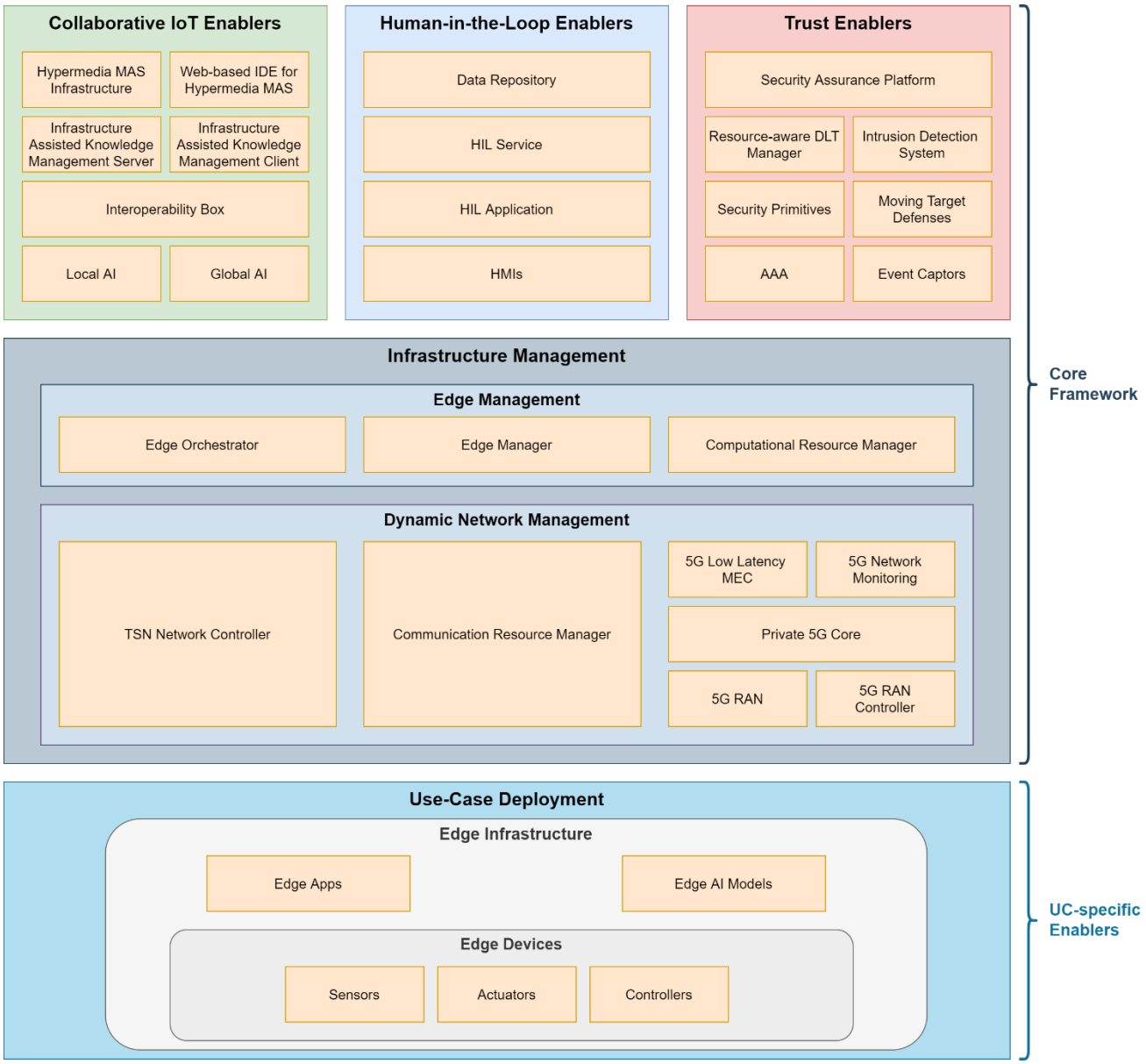

Figure 13. High-level view of IntelIIoT's logical architecture

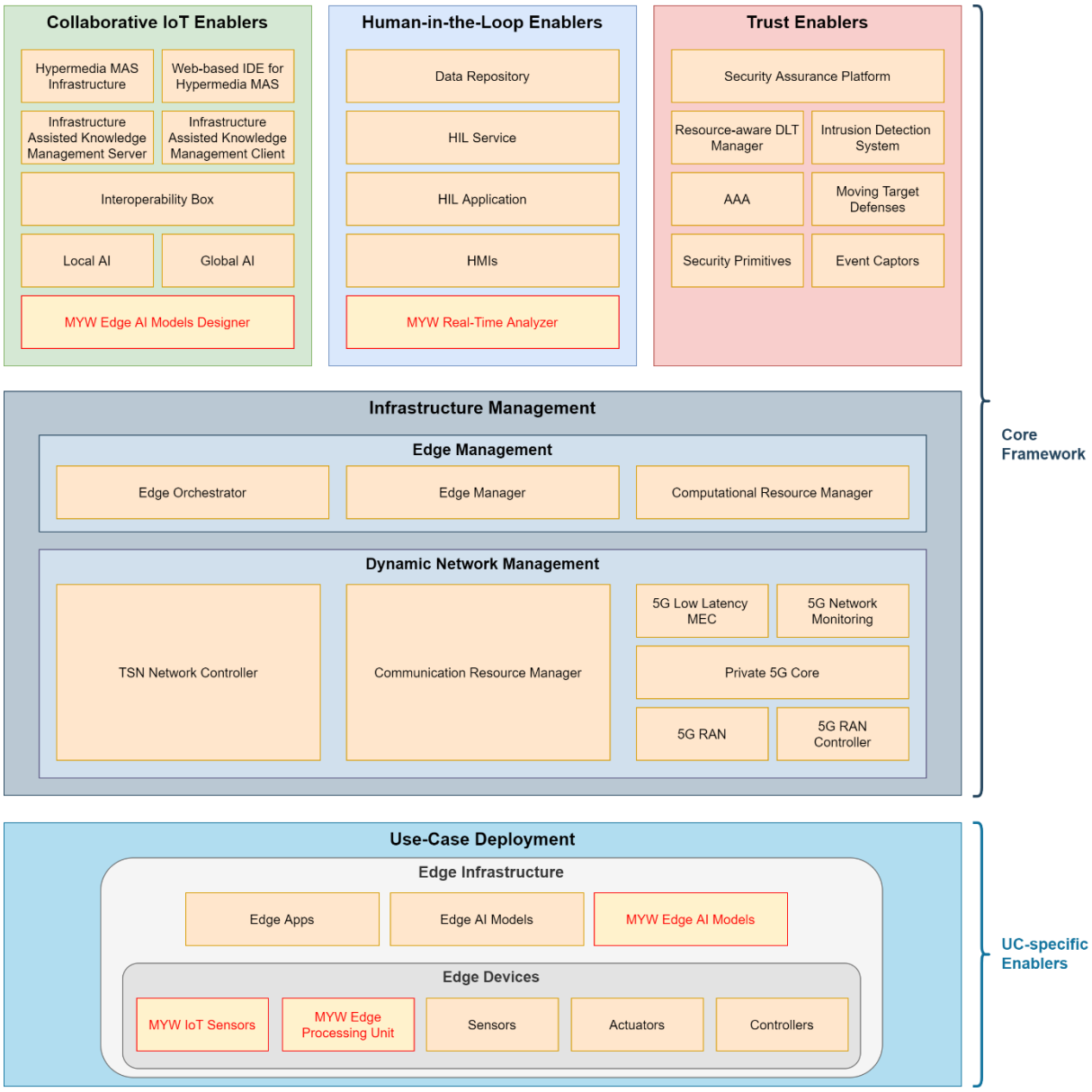

Figure 14. High-level view of IntellIoT's logical architecture with Open Call 1 contributions to core framework enablers

2.1 Logical View

The Logical View *(how the system is structured)* per IntellIoT component group is provided in the subsections that follow.

2.1.1 COLLABORATIVE IOT ENABLERS

The Collaborative IoT pillar of IntellIoT focuses on the flexible interaction of edge apps with the hypermedia-based agent infrastructure and on the distribution of AI components across the system and within the system's components. In the following, we first discuss the system-wide AI components that are responsible for the high-level management of edge apps (e.g., tasking an edge app with a specific task), which are shown in Figure 15, and then those AI components that are deployed on individual IntellIoT components and their global support

infrastructure (e.g., the local AI on a robot or tractor together with the federated learning components that support these), shown in Figure 16.

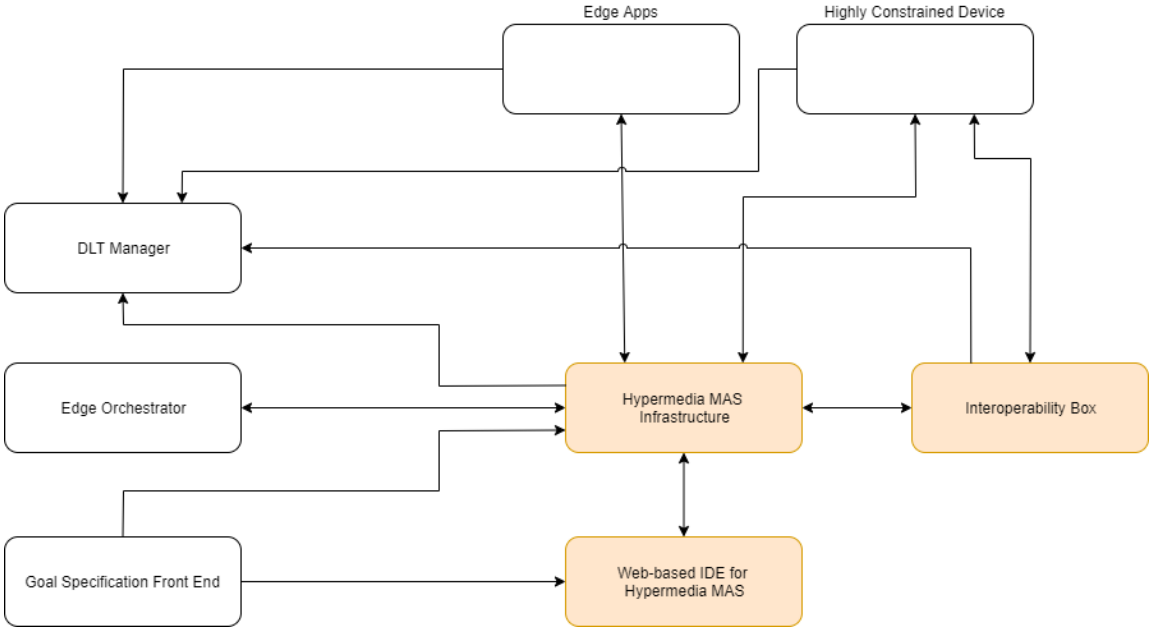

Figure 15. System-wide components that are responsible for the high-level management of edge apps.

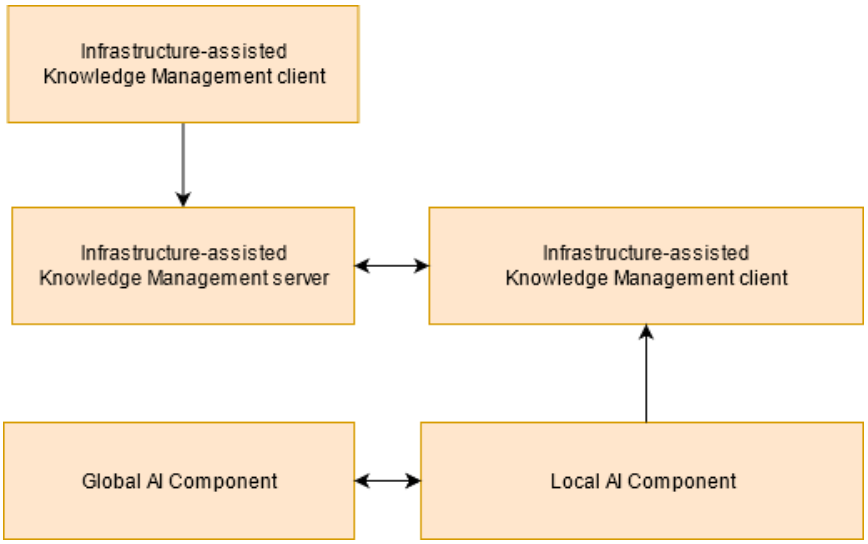

Figure 16. AI modules deployed individually on IntelIIoT components.

System-wide Components

The kernel of the system-wide AI components in IntelIIoT is formed by the **Hypermedia Multi-Agent System (Hypermedia MAS) Infrastructure**. This component runs a multi-agent system (i.e., it hosts several agent programs that are arranged in an organization, and each have their own procedural knowledge in the form of trigger-action rules) and reacts to incoming goal specifications (e.g., "Field 6 should be harvested" in UC1 or "Wooden plate should be engraved with the text …" in UC3) by executing towards this given goal. The Hypermedia MAS Infrastructure is configured by a farming/health/industrial engineer using the **Web-based IDE for Hypermedia MAS**. In this component, the engineer – given a goal scheme – specifies an agent organization and

agent procedural knowledge that considers available edge app functionalities and that together are able to reach the defined goal. To enable this, the IDE monitors the Hypermedia MAS Infrastructure for available services that provide W3C WoT Thing Descriptions (TDs) or W3C WoT Thing Description Templates (TDTs), and the engineer configures agent procedural knowledge (e.g., a trigger-action rule that governs the behaviour of an agent) on top of these interface descriptions. This mechanism avoids tight coupling of agent programs with specific edge apps or devices and ensures (among other features) that functionalities are provisioned at run time through the Edge Orchestrator. An important prerequisite for this is that edge apps register their TDs or TDTs with the Hypermedia MAS Infrastructure. At run time (i.e., after the Hypermedia MAS Infrastructure has received a goal and is executing its MAS towards achieving this goal), the agents render interface calls towards the edge apps given the TDs. If a TD is not available yet (i.e., if an agent encounters a TDT without protocol binding), the agent requests this edge app to be provisioned by the Edge Orchestrator. After this process, or when a protocol binding is available from the beginning, the agent directly interacts with the respective edge app.

To enable the integration of highly constrained devices, the **Interoperability Box** is a software component that aims to provide TDs and interfaces to devices that do not have edge apps managing them. The Interoperability Box supports several different networking technologies, such as Radio (e.g., 802.15.4) and Wifi (802.11x), and protocols (MQTT, HTTP, etc.) for device connectivity. For each available device, the Interoperability Box provides a TD to the Hypermedia MAS, and creates an interface for agents to communicate with the device. The Interoperability Box could be considered a generic edge app that supports multiple types of highly constrained devices and can manage multiple such devices in the same instance.

Linking to the Trust Enablers of the IntellIoT project, the Agents within the **Hypermedia MAS Infrastructure**, the **Edge Apps**, and the **Interoperability Box** connect to the **DLT Manager** to enable the decentralized journaling of the system's operation. This is accomplished by these components, registering: (1) the tasking of Edge Apps and of the Interoperability Box (by Agents); (2) the tasking of constrained devices (by the Interoperability Box), and (3) requests by agents (by Edge Apps and the Interoperability Box).

Individually Deployed Components

Creating knowledge from advanced machine learning (ML) techniques requires significant resources, which makes the simultaneous creation of the same type of knowledge highly inefficient. Similarly, to the human education system, knowledge does not need to be recreated but can be shared. This becomes possible with the **Infrastructure-assisted Knowledge Management (IAKM)** module for the proposed decentralized cooperative AI mechanisms in the three defined use cases. An IAKM *server* is located in the IntellIoT infrastructure (private network or private 5G MEC), and its role is to act as AI/ML broker and storage according to W3C-defined semantics. An IAKM *client,* whose role is to interface to the resource-aware re-trainer component and to connect to the IAKM server (located at the 5G MEC or further in the private cloud) using REST and HTTP over TLS (HTTPS) technologies, respectively. The IAKM client is deployed on each edge device. The IAKM first sets interoperable semantics uniquely defining the meaning and usage of a ML model, then proposes a publish/subscribe mechanism to share or acquire a model that an agent requires from another agent.

Under the federated learning setting, the **Global AI Component** acts as the centralized entity that supports individual devices to share the training results of their local Manufacturing, Healthcare, and Agriculture AI Models with one another while at the same time maintaining local data private. The Global AI Component is located on a central server, and it manages the entire federated learning process. This component uses HTTP over TLS[2] (HTTPS) for the communication with local devices and plays two major roles. First, as model aggregator: AI model updates from local devices are sent to the Global AI Component, which then aggregates these updates to create an updated model. The updated global AI model is evaluated and, pending improved model performance and considering model personalization, is shared with local devices. This enables sharing the local knowledge from each device with one another. Second, as resource-aware retraining considers available computation and communication resources to schedule local devices that need to contribute for re-training tasks. The information on the available resources scattered over the network is delivered with the aid of the IAKM system.

The Global AI Component must be configurable in terms of number of local devices that can participate, AI model parameters and aggregation methodologies. It can also access details of the local devices from a database at the

[2] https://datatracker.ietf.org/doc/html/rfc5246

start of execution, such as identification number and domain-related use case. Different AI model aggregation methods deal with the communication overhead caused by the model update process, but the discrepancy between the actual and compressed versions of the updates can practically lead to AI model divergence. Some methods are very effective in managing this but can contain more than twenty-five million coefficients with possible size of a serialized AI model of 100 MB. The strategies also need to maintain local device state and require frequent communication with the Global AI Component. It means that the Global AI Component must provide a way to port almost arbitrary compression schemes and collect related metrics.

The **Local AI Component** is responsible of training AI models using the local datasets at the robot, patient, or tractor devices. Such training allows local AI models to capture the correlations between locally observed data and corresponding control decisions, as well as running inference on control decisions at runtime. This leverages the collective knowledge and possibly produces improved AI models. The Local AI Component communicates with the Global AI Component over HTTPS and sends the updates that result from training local AI Models.

The federated learning process consists of training rounds. For each training round, the Local AI Component publish model weights along with other information to the Global AI Component. They also let the Global AI Component know that each is up and running. The Local AI Components can also publish the size of the local dataset, which can be used to determine the most valued minimum number of local devices and to determine the bias during aggregation.

Each use case will need domain or task-specific Agriculture, Healthcare, and Manufacturing AI Models. As described previously, base AI models will be hosted in the Global AI Component and local models in the Local AI Components. The **Agriculture AI Models** will provide intelligence to the tractors to overcome the obstacles in their path. The models will be trained in a supervised manner using a camera image-based training dataset along with human-operator decisions with the focus of driving around obstacles. The **Healthcare AI Models** will be used to predict different clinical outcomes of heart failure patients. It is envisioned that models can be implemented for patient monitoring and diagnostics as well as for recommendations, especially as related to exercise. The **Manufacturing AI Models** will provide intelligence to the manufacturing machines to handle the work pieces. The models will be trained in a manner similar to that used for the Agriculture AI Models. Since the models will be trained within a federated learning system, the degradation of deployed models will be prevented, and outlier measurements can be addressed. Possible model architectures that will be implemented within IntellIoT include neural-backed decision trees, which address the issue of model interpretability, and Long Short-Term Memory networks, which are suitable for measurements taken in a time sequence. Traditional probability models will also be employed as appropriate.

The personalization of AI models especially in the Healthcare use case brings challenges due to unbalanced and non-identically distributed data that may damage loss convergence and reduce model robustness. To handle such problems, the federated learning system will have "local adaptors". Globally trainable weights will specialize in extraction of task-specific features from the data after the training, whereas locally trainable weights learn specific transformations of those features. Only trainable parameters are shared among the Local AI Components, while the normalization statistics remain private during the training. In this system, the global AI model can achieve baseline performance, but discrepancies may arise between the local AI model parameters. To avoid the discrepancies, the system must support the persistent model state and allow the categorization of model parameters into sharable and private as well as allow an evaluation of the local AI models. During inference, Local AI Component can decide to infer locally using its local AI model to get aid from the Global AI Component to improve its inference accuracy accounting the limitations in computing and communication resources.

Open Call 1 contributions

As OC1 partner, MYWAI is contributing to the Collaborative IoT pillar of IntellIoT through the **MYW Edge AI Designer** module, represented in Figure 17.

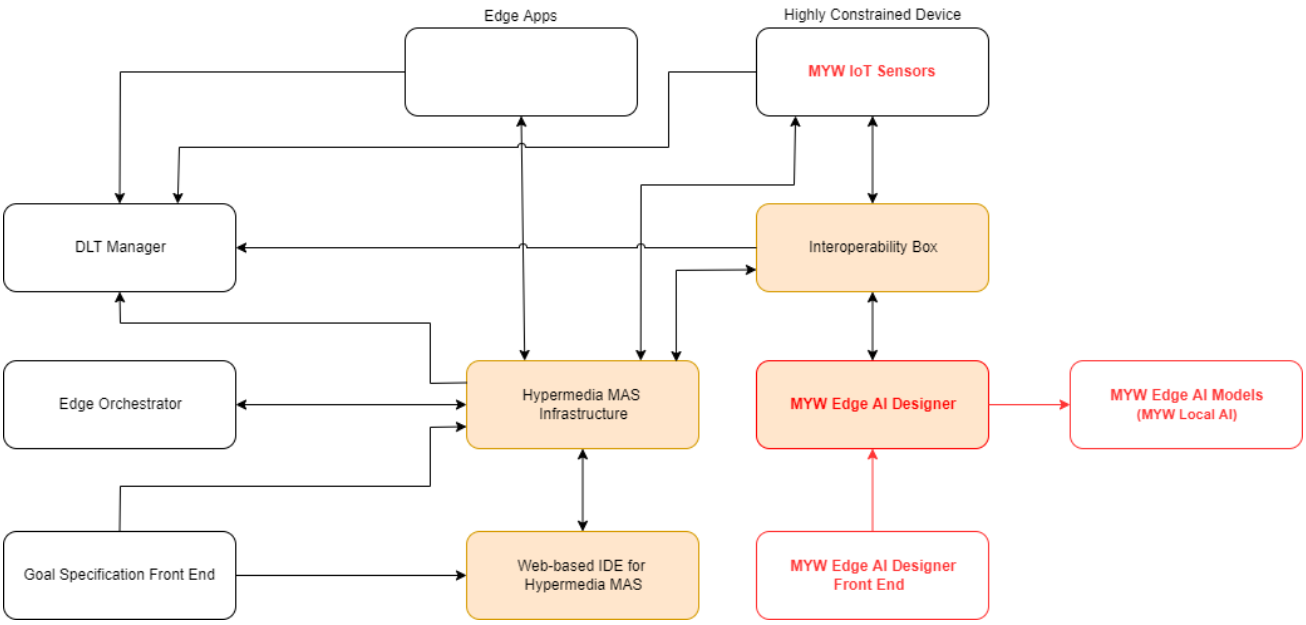

Figure 17. Logical view of the MYWAI Collaborative IoT Enablers

The **MYW Edge AI Designer** component aims at supporting the full workflow for designing, developing, and training global ML/AI models as well as the deployment of local AI suitable to run on edge devices. The MYW Edge AI Designer exposes a front end that can be used by use case domain experts to access a set of functionalities enabling the visualization of data gathered from sensors as well as the detection and labelling of data subsets representing either normal conditions or anomalies and deviations, as a reference for the development of ML/AI models. Through the MYW Edge AI Designer the domain expert will also be able to access hierarchical and structured information about the monitored target (e.g., tractor, robot arm) and its sensors, through the adoption of the interoperable OPC UA specifications wherever and whenever possible/relevant.

Provided that the MYW Edge AI Designer can be used to build global and local AI from data acquired by virtually any sensor offering an interoperable interface to get the data, MYWAI can make available for integration in IntellIoT own **MYW IoT Sensors** (i.e., IoT edge devices specifically conceived and designed to monitor specific types of conditions in the physical world and suitable to be used in the IntellIoT UCs.) Two different MYW IoT Sensors are considered: MYW Smart Tag (3-axis accelerometer, Gyroscope) & MYW Tracking Device (3-axis accelerometer, Gyroscope, Temperature, Humidity, Pressure, Gas/smoke, Geographical position via GPS GLONASS Galileo).

2.1.2 HUMAN-IN-THE-LOOP ENABLERS

The Human-in-the-Loop (HIL) enablers handle the reactive approach of interaction, enabling human oversight and intervention when necessary. This is important not only from a practical perspective (e.g., when the machines do not know how to first handle a situation), but also from a trustworthiness perspective, as Human agency and oversight is one of the key requirements towards Trustworthy AI [2].

Taking IntellIoT's Use-Cases as examples, a number of HIL enablers, as shown in Figure 18, have to interact with one another in order to achieve the Use Case goals. The next paragraphs will explain these interactions in more detail.

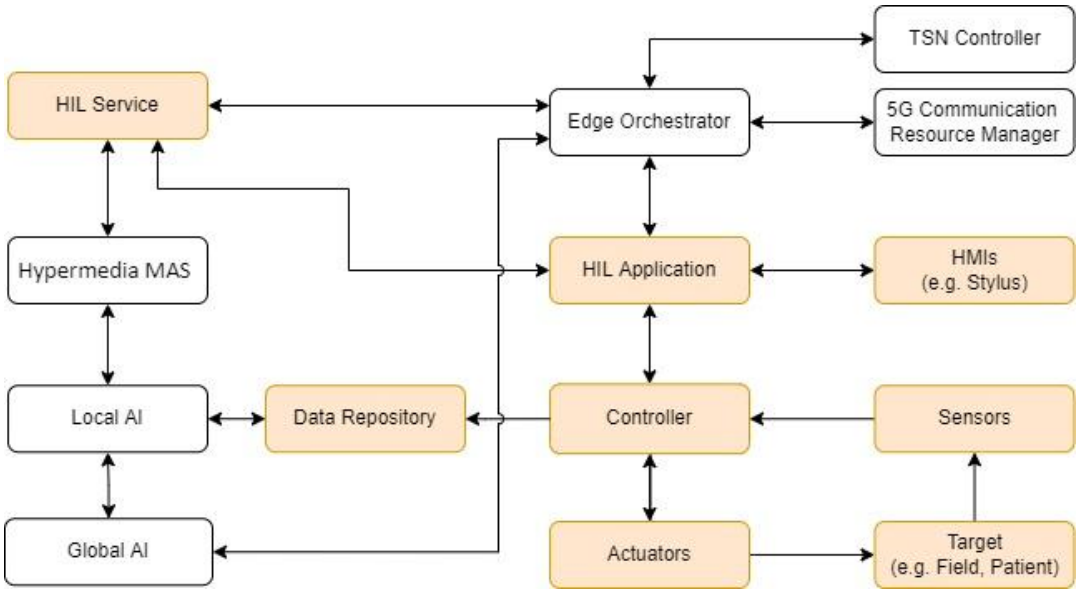

Figure 18. Logical diagram of Human-in-the-Loop.

To understand the HIL logical diagram, one must begin with the **Target** component. Manipulating this in some form is the IntellIoT framework's end goal. Whether it is a field that a tractor must autonomously navigate and use its tools on, a patient's health status which is to be monitored and improved if necessary, or a wooden workpiece that needs to be engraved, the logical diagram remains largely the same.

Sensors provide information about the target. These can be in the form of cameras, biometric devices, etc. For example, in UC3 a camera is used as the sensor, mounted on the head of the robot. Additionally, robot-internal sensors are used to provide information about the current state of the robot. Both the video stream from the camera and the robot state information are provided to the **HIL Application** (but further explanation of that later on).

The **Controller**, physically realized as an edge device, can host multiple applications with a wide range of tasks, the main one being, of course, the manipulation of the target through the system's actuators. This, of course, varies from use case to use case. For UC1, the Controller for the Tractor will be considered an edge device mounted on it. The tractor controller will host multiple applications (e.g., the DLT client, MTD Client, Tractor AI, IAKM Client and potentially have the 5G connection) to either interact with other entities in the field and with the human operator. Additionally, the control applications for the tractor (e.g., steering, driving, acceleration) will also be hosted on the edge device and will directly control the vehicle. For UC2, the controller will take the form of the smartphone devices, where edge applications (e.g., for monitoring wearable devices, for local analytics) will be deployed. For UC3, the edge devices run multiple controller apps, e.g., the robot controller, machine controller, TSN controller. Additionally, multiple applications (e.g., the DLT client, MTD Client, AI, IAKM Client) are deployed on the edge. Due to the stringent requirements on timing and reliability, some applications, e.g., the machine controller functions, may require to be deployed on a dedicated hardware close to the machine. However, as long as the aforementioned requirements are fulfilled, deployment on any edge device is possible.

The controller can receive commands to control the actuators either from the **Hypermedia MAS** or the **HIL Application**. For example, the robot controller will receive transport requests from the Hypermedia MAS, and direct movement commands from the HIL Application.

The actuators of the system can vary. In UC1, the actuator can be considered the tractor, in UC2 the actuators can be loosely considered as the recommendations and interventions to the patient, and in UC3 the actuator is the robot itself.

Now consider the case of autonomous operation of the system's controller, which in turn controls the actuators, and the actuators manipulate the target. The sensors complete the feedback loop, and the process continues. Here, it is the **Local AI** that constantly needs to make decisions on what commands to send to the controller. At some point, it is possible that the Local AI will be unable to decide with high confidence, and if the **IAKM component**

cannot provide assistance, it will need to request the assistance of the human agent, thus keeping the *Human-in-the-Loop*.

The role of the **HIL service** is to connect an AI requiring human support (e.g., the AI to compute a grapspot for the robot or the AI on the tractor) with an entity being able to provide such support. The HIL Service is triggered through an escalation by the AI component (e.g., in case the AI is unable to find a possible way to solve a task) that is brokered by an Agent. This Agent is running in the Hypermedia MAS Infrastructure and has been programmed by the user of the Web-based IDE for Hypermedia MAS to handle HIL escalation requests and invoke the **HIL Service**, which requires that the functional description (i.e., the W3C WoT Thing Description) of the HIL Service is available to the Hypermedia MAS. To enable this brokerage, the HIL Service maintains knowledge about availability, reachability, and relevant properties of available **HIL Applications**. Upon receiving a support request, it contacts one or more available HIL Applications. The connection is established through accepting the request, by the human operator. Only one Human Operator can establish this connection. Where stringent requirements on timing apply, it initiates the reservation of communication services. Therefore, it contacts the **Edge Orchestrator** to update the traffic demand for the HIL service, which requests further resources at the **TSN controller** and the **Communication Resource Manager**. The Edge Orchestrator also takes care that the availability constraints of the HIL Application are met. Therefore, it continuously considers starting and terminating HIL Application instances. While the above hold for UC1 and UC3, in the healthcare use case (UC2), due to the sensitivity of the application, a less automated approach is followed, whereby the main HMI is the smartphone application and more asynchronous & active involvement from the human (i.e., clinician) is needed.

In terms of the infrastructure management enablers supporting the above, the **TSN controller** gives guarantees for the QoS of the embedded communication service (if possible), and configures network nodes, so that tactile interaction between a human operator and the system is possible and reliable. Furthermore, summoned by the Edge Orchestrator, the **Communication Resource Manager** opens, configures, maintains, and monitors dedicated 5G resources required by the HIL service.

The **HIL Application** is a component that is not pertinent in all use cases investigated in the project, yet it remains a central component of the ones in which it is involved (UC1 and UC3). Before addressing its core functions, one must first understand that all human interventions take place through the use of **HMIs (Human Machine Interfaces)**. These HMIs will vary depending on the use case. In UC 1, the human in the loop, also called Human Operator, will interact with the machine through a **VR headset** (Oculus Quest 2) and its **controllers**. In UC2, the main HMI will be different types of **Smartphones**, such as an iPhone or Android. Both Patients and Doctor will make use of them, though their inherent tasks are very different. Each one will have their own app interfaces allowing the required feature. In UC3, the human will interact with the machine through an **AR headset** (Microsoft HoloLens 2) and a pen –like **input device** (Holo-Stylus). Both the VR and AR headset, as well as their respective input devices, are connected to the HIL Application. In case of UC 2, the devices will be connected either to the Data Repository, or the Controller. More details about this are given in the section dedicated to the process diagram. The HMIs also encompasses additional human-interfacing capabilities of IntellIoT, such as the Goal Specification Front End, the Security Assurance Platform Front End, etc., that are integral to other enablers of the framework.

In a nutshell, the **HIL Application** is necessary when the HMIs involve the VR and the AR headset. AR and VR devices are considered *edge devices*. Their most impacting limiting factor is the processing power. There is a tricky design decision that manufacturers must make: a small processor built into the wireless headset, or an external processor wired to the headset. The first option is comfortable to wear but cannot render complex 3D models (several millions of polygons), simulations, or other performance heavy content without a significant drop in performance and frequent crashes of the device. Because these devices are worn like glasses, these performance drops can be at best very uncomfortable for the user, or at worst cause headaches or nausea. The latter allows the rendering of complex files but limits the movement and the cables can make them uncomfortable to wear. In short, both versions have their merits, but neither solves all problems. Besides the issue with processing power, interfacing with UWP (HoloLens specific issue) is difficult with some libraries or simply impossible with others. Both issues needed to be addressed during the concept creation phase. This project requires an application that can handle a steady video stream of a live camera feed in real time, the display of several 3D models, interactable UI elements, as well as the reception and transmission of robot controls and metadata. Furthermore, it has to be integrated with various interfaces. Hence, Holo-Light will build a solution that is based on its own Remote Application Rendering technology (called ISAR). The HIL Application is one component of the **two-component solution**. The other

component is a use-case specific client (in the IntellIoT demonstration this is one for Use Case 1 and one for Use Case 3) – and will interface with the Controller either directly or through WebRTC. The **HMIs** connect to the HIL Application via the second component – the client, a simple xaml based application installed on the device. There is no direct connection between the HMIs and the HIL Service. Additionally, in Use Case 3 (Manufacturing) the HIL Application integrates with the **Stylus SDK.** The data transmission will occur via the **5G Network** that links up with the **TSN Controller**. A direct interface with the **Local** or **Global AI** has to be determined at a later stage of the project.

Shifting the focus away from the HIL Application and the HMIs, all human interventions, as well as data coming from the sensors, are aggregated by the controller on a **Data Repository**. In the healthcare use case (UC2), the Patients **Data Repository** will store only measurements (patient data) that are required for making the reports and the recommendations that need to be sent to the physician (Human-in-the-Loop). The data models of the repository will be standardized to comply with relevant standards (e.g., FHIR). In use cases 1 and 3, System Data Repository is used to store the local sensory information at the device (tractor and robot); these data will be used for local AI model training as well as inference.

As an HIL enabler, ***Local AI*** reuses the human control commands as new labelled data to re-train its local AI models. Once model evaluation at the Global AI Component shows improved model performance, taking into consideration also the personalization of local AI Models, an updated AI Model can be shared with all devices within the system, enabling the transfer of the new knowledge that has been collected from the human intervention. In this way, the human-in-the-loop and the IntellIoT framework make it possible to use the human control commands to train the AI models further with this new labelled dataset, which is otherwise extremely scarce and difficult to obtain.

Open Call 1 contributions

As OC1 partner, MYWAI is contributing to the Human-in-the-Loop pillar of IntellIoT through the module **MYW Real-Time Analyzer** introduced in Figure 19, and representing and additional HIL Application proposed by MYWAI for detecting and notifying to an operator anomalies and faults affecting a target (e.g., tractor, robot arm).

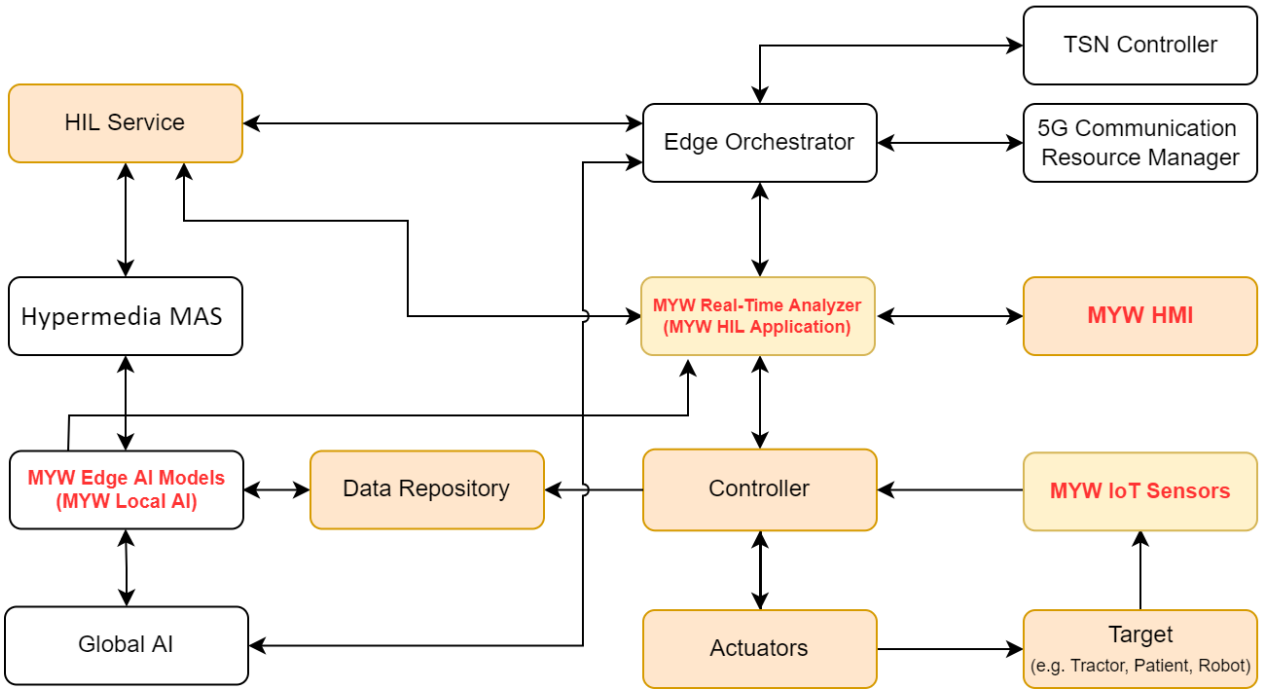

Figure 19. Logical view of the MYWAI HIL Enablers.

The **MYW Real-Time Analyzer** component is aimed at supporting the human-in-the-loop by notifying anomalies/faults inferred by the local **MYW EDGE AI models** running on suitable computational devices at the edge/very edge and able to automatically process data gathered by sensors in the specific application context Functionalities for retrieving and visualizing sensors data and real-time equipment information and status are

made available to the operator in charge of monitoring the target through a user-friendly GUI (**MYW HMI**) accessible from different devices (e.g., tablet, smartphone, local monitoring console, etc.).

2.1.3 TRUST ENABLERS

The Trustworthiness pillar of IntellIoT (Figure 20) focuses on addressing the security, privacy and trust needs of the framework. As such, it features mechanisms allowing automated and continuous security and privacy assurance assessments, whose purpose are to enable trust of the human in the system. These are, therefore, aligned with Objective 4 of the project (*"Enable security, privacy and trust by design with continuous assurance monitoring, assessment and certification as an integral part of the system, providing trustworthy integration of third party IoT devices and services."*).

Security, privacy, and trust are considered early in the design phase, integrating mechanisms for the system to proactively defend against prominent threats (e.g., by continuously modifying its network topology and/or configuration, making the planning and execution of an attack much more difficult). Reactive mechanisms are present as well, involving trust-based mechanisms that quantify and compute trust, detect anomalies in the system through the variability of said trust and deploy the appropriate mitigation measures to isolate malicious or compromised entities. All these mechanisms are supported and enabled, as needed, by continuous monitoring and evidence aggregation methods, which provide an up-to-date view of the security and privacy posture of IntellIoT's deployment and the associated use case infrastructure to the human operator.

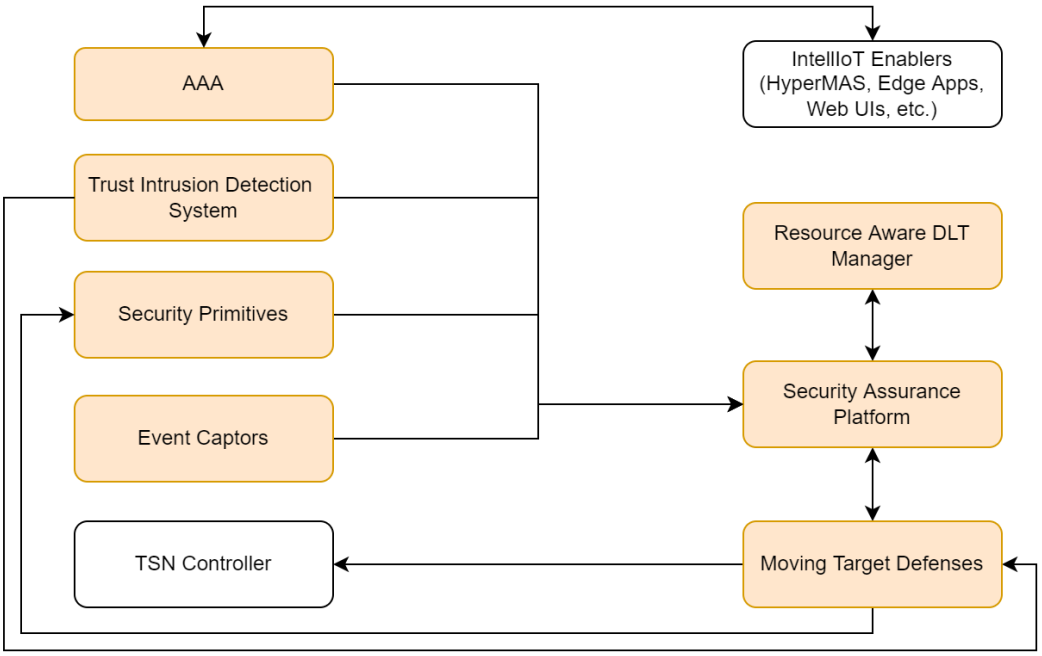

Figure 20. Trust enablers logical diagram.

Central to the trustworthiness components is the **Security Assurance Platform (SAP)**, an integrated framework of models, processes, and tools to enable the continuous assessment & certification of the security properties of the system. It uses different types of evidence to demonstrate the support for the required properties and produce the corresponding certification assessment. This evidence includes monitoring and testing data that are provided from external tools to which the platform is programmatically hooked to (e.g., the various purpose-developed Event Captors and other security mechanisms, such as the Trust-based Intrusion Detection System). Additionally, the SAP is responsible for triggering the appropriate mitigation measures (via interaction with the Moving Target Defences) when the aggregated evidence points to a potential attack being carried out. Furthermore, the SAP assists in recording important trust-related events (e.g., a security posture change) by forwarding the relevant records to the Distributed Ledger Technology (DLT) manager.

The **Event Captors (ECs)** mentioned above are software modules responsible for aggregating pertinent evidence from multiple sources related to the operation of individual components, as well as the overarching processes where these components are involved in, thus enabling the real-time, continuous assessment of the security posture of the IntellIoT system. ECs can be deployed across all layers of the IntellIoT architecture, from edge devices (e.g., the robot arm, to monitor telemetry that may indicate abnormal activities) to device operating systems (e.g., to monitor running processes) and backend storage databases (e.g., to parse access logs or calculate uptime), transmitting their monitoring evidence to the SAP.

The **Trust-based Intrusion Detection System (Trust IDS)** is a software component that can also be considered as an event captor, in the sense that it continuously records events: it monitors the network traffic and analyses it for anomalies, for example excessive throughput from another network node (e.g., any IoT device or system participating in the network). Based on the analysis, it computes a local trust value for each other node that it communicates with. When the trust value for a node drops below a threshold, it generates a warning. These warnings are left to be processed by the rest of the system in order to determine if an action is needed to be taken against offending nodes.

Moving Target Defences (MTDs) are a client-server set of software components that manage the network configuration of the Edge network. The MTD server proactively, at fixed intervals, generates new configurations in order to invalidate any information about the network an attacker might have procured. Configuration changes will focus on network layer 3 and above, thus the TSN controller and the resource reservations it maintains are not affected by these changes. In case of an active attack, the MTD server can be configured to either: (i) automatically generate a mitigation configuration based on predefined strategies, or (ii) receive an action (e.g., isolate node X) from the HIL through the Security Assurance Platform. The MTD clients are responsible for applying these configurations. As part of the MTD, traffic inside the Edge network is encrypted and transmitted through tunnels (PPTP with IPsec), in order to avoid packet sniffing and Man-In-The-Middle attacks. The **TSN controller** which manages the wired network resources and, through its northbound interface, allows the MTDs to request for the exclusion of dedicated end devices from any communication with the TSN network. To achieve that, network controller will make use of features of the underlying network infrastructure, e.g., VLAN configuration or PSFP filtering.

The **Authentication, Authorisation, and Accounting (AAA)** component provides user, user role and key management, as well as an authentication mechanism for external applications that need to access internal services of the IntellIoT framework. It covers both Web UI type of logins as well as automated login using trusted clients. Trust enablers communicate through a separate secure channel provided by a common broker.

The **DLT Manager** includes a distributed ledger and a layer-2 off-chain storage solution. In particular, DLTs provide a decentralized, transparent and immutable solution for recording the events and activities in systems. However, recording raw data in the ledger requires too much processing speed and reliability, so that a layer-2 off-chain storage solution should be included to store the raw data and the hash version of data. The DLT Manager is responsible for accounting and auditing activities from the Hypermedia MAS, the Edge Apps, the TSN controller, and communicates with the Security Assurance Platform for verification of the trusted data.

In addition to the above, Security Primitives are also utilised as part of the above solutions to strengthen the overall security of the system by design. Indicative such primitives integrated across IntellIoT include:

- Cryptographic Operations as part of the MTD solution, that encrypt and optionally sign traffic between clients, and transfers it through Point-to-point tunnels.
- Secure Communications since all connections use the TLS protocol or similar.
- Account Authentication and Management, as part of the AAA component.
- Anomaly detection and reaction, provided by the ECs and IDS components.
- Cryptographic Key and Certificate store, as part of the AAA component.
- System Event Logging, as part of the SAP and DLT components.

Open Call 1 contributions

IKH developed a Farmer's logbook enabling the farmers to log securely their actions and also supporting automatic saving of events through e.g., the robot perception functionality, through the DLT manager.

2.1.4 INFRASTRUCTURE MANAGEMENT

The communication and computation infrastructure (Figure 21) builds the infrastructure foundation of the IntellIoT framework and allows the deployment and dynamic management of edge applications (**Edge Apps**). It is thereby the foundation for achieving Objective 1, to "Create a self-aware and semi-autonomous multi-agent system over an optimized computation and communication infrastructure that manages compositions of IoT/edge devices in closed-loop with the network".

The infrastructure management as shown in Figure 21 and discussed below largely remains as defined during Cycle 1 and presented in D2.3. However, there is no need for the NFV Controller anymore, as no NFV functions are being used in the project, hence it has been removed.

The integration approaches of the open call partners with the IntellIoT framework do not affect the infrastructure management, as none of the open call partners provides core infrastructure management components. In fact, both Trilogis and MY.WAI rely on infrastructure components which are hosted in Cloud computing environments.

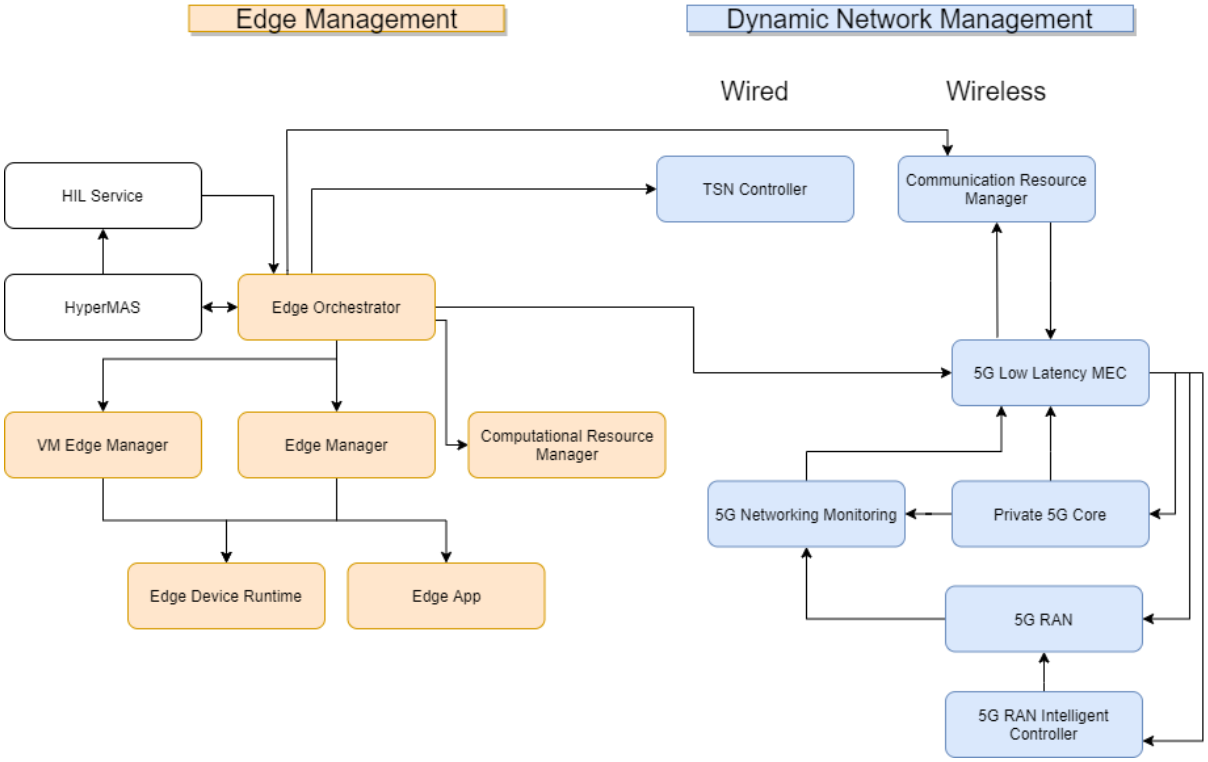

Figure 21. Logical diagram of the Infrastructure Management.

The deployment of an Edge App is triggered from the **Hypermedia MAS** (see Section 2.1.1) when a specific service is required. The central point for triggering the deployment of Edge Apps is the **Edge Orchestrator** that works together with one / multiple **Edge Manager(s)** to orchestrate the efficient deployment of an Edge App onto one / multiple edge devices. An edge device is an off-the-shelf entity that hosts an **Edge Device Runtime**, which is able to run Edge Apps. The Edge Device Runtime is controlled by the Edge Manager. An edge device with an Edge Device Runtime must be on-boarded through the Edge Manager. The Edge Orchestrator optimizes the allocation of Edge Apps on the edge infrastructure. Therefore, Edge Apps are deployed and moved inside the edge infrastructure, maintaining the application state. The Edge Orchestrator dynamically determines a schedule for allocation taking the network condition into account. The deployment of Edge Apps is executed by the Edge Manager. Therefore, the Edge Manager provides a service interface for managing edge apps and edge devices. There are applications that cannot be bundled into an Edge App, as they are constrained to certain execution environments (e.g., the HIL

application has to run in a VM). The **VM Edge Manager** is responsible of controlling the deployment of those applications.

The Edge Orchestrator uses the **Computational Resource Manager** to determine optimal allocations. It comprises an algorithm for the optimized allocation of individual Edge Apps or parts of Edge Apps composed of functions onto edge devices, while considering the network configuration. The deployment of Edge App functions can be tuned towards different optimization goals, e.g., reliability of compute nodes, response time of the application, or energy consumption. The Computational Resource Manager takes the IoT application configuration, its constraints, as well as the current edge and network configuration as input. After determining an optimal allocation, it returns a specification of it as output. During the whole allocation of an edge app, the Computational Resource Manager monitors whether optimality criteria for Edge Apps are still met.

The **Communication Resource Manager** is responsible for dynamically reconfiguring the 5G network (RAN and CN) for optimal performance of the 5G infrastructure, i.e., to open, configure, maintain and monitor dedicated 5G resources. For this, it receives communication service requests from the Edge Orchestrator and provides network monitoring and resource availabilities to it. The optimizations and re-configurations of the 5G network can be based on network data, e.g., performance or channel measures metrics obtained from the network. The aim is to support heterogeneous communications (e.g., ultra-reliable low-latency, massive IoT traffic, VR/AR video…) between IoT devices, humans, and services and fulfilling the individual requirements of each traffic type.

The **5G Low Latency MEC controller** (LL-MEC) system is responsible for low latency 5G services at the private 5G MEC. It is composed of a 5G Flex Core Network controller (FlexCN), which can dynamically alter private 5G Core services, as well as MEC service management for edge apps (also called xApps in OAI). MEC technologies are critical to Objective 2 of the IntellIoT project, as it enables at the same time low latency IoT services through shorter loops between edge devices and infrastructure components, as well as responsive IoT services, by dynamically adapting to 5G conditions. Finally, MEC technologies maintain data local and private. The LL-MEC receives radio & service configuration requests from the Communication Resource Manager and connects to the 5G Core to send MEC instructions, to the 5G RAN controller to trigger radio resource reservation, to the 5G NVF controller to allocate computational and storage resources, to 5G RAN to extract RAN data required for MEC services, and finally to 5G Network Monitoring component for MEC monitoring and analytics.

The **Private 5G Core** offers "core network" (CN) functionalities of the 5G infrastructure. It is considered as private, as the CN is hosted as a private network and may keep corporate data and knowledge locally. The Private 5G Core includes cloud-native functions (CNF) implemented as microservices, and as such, the Private 5G Core operates as a multi-microservice component. The Private 5G Core provides the basic 5G functions for authentication, routing of data transfer between IP networks and wireless devices via gNB base stations. It also handles routing of third-party services (e.g., edge orchestration, etc..).

The **5G RAN** component is responsible for providing 5G URLL/eMBB communication between devices and the 5G infrastructure. The 5G RAN is composed of a 5G gNB as well as a 5G UE and implements a 5G FlexRAN API required by the 5G RAN controller. The 5G RAN will also implement 5G ProSe D2D Sidelink communication between two (or more) 5g UEs.

The **5G RAN Intelligent Controller** is responsible to configure the 5G radio resources that the 5G RAN component will provide. The 5G RAN intelligent controller triggers the 5G Network Monitoring component with network quality indicators. The 5G RAN controller exposes a RAN descriptor (RD) API to the Communication and Resource Manager component.

The **5G Network Monitoring** component is in charge of monitoring the various KPIs that are used by the various 5G components (RAN, NVF) for optimization. The 5G Network Monitoring component, based on 5G RIC APIs, is particularly important for operators to understand the reason for a particular optimization or limitation. For example, it monitors the 5G resources, in particular the 5G slice resources. It monitors the 5G network parameters, such as the reliability or the delay. It can monitor the edge and device resources; the 5G Network Monitoring component also operates as a Logging agent and monitors the various messages exchanged between the various components. Accordingly, partners responsible for a particular component can correct and optimize its behaviour remotely. The 5G Network Monitoring component is finally in charge of displaying the status of the various components, either being active, inactive or other state that the component would provide.

The **TSN controller** receives communication service requests via its northbound interface. Example for a requesting entity is the **Edge Orchestrator** in UC3. TSN controller checks the requested QoS attributes of a communication service request for feasibility on the active network topology and gives guarantees for the QoS of the embedded communication service if possible. To do so, it reserves resources in devices of the underlying deterministic network infrastructure and changes configurations of the latter if necessary. It continuously monitors the active network topology and reacts on changes, e.g., added devices or faults, with the goal to keep given QoS guarantees wherever possible. The TSN controller interfaces with switches and end stations within the deterministic network infrastructure via its southbound interface. The southbound interface is based on Netconf protocol and is described in YANG models. Particularly relevant for TSN are the YANG models described in IEEE802.1Qcw. Additionally, SNMP-based or proprietary interfaces are commonly used in practice.

2.2 Processes View

The Process View (*how the system behaves*) per IntellIoT component group is provided in the subsections that follow.

2.2.1 COLLABORATIVE IOT ENABLERS

In the following, we specify the main processes between the components of the Collaborative IoT Enablers. The main processes that are relevant within the IntellIoT Collaborative IoT pillar, also depicted in Figure 22, are:

- **Configuration**: The Domain Engineer (e.g., manufacturing engineer, farming engineer) configuring the Hypermedia MAS
- **Execution**: The Domain Expert (e.g., customer, farmer) specifying a goal towards the Hypermedia MAS
- **Edge Orchestration**: The Edge Orchestrator providing a W3C WoT TD at run time
- **Retraining**: The ML Subsystem retraining and publishing a new model
- **Model Update**: The ML Subsystem updating to a newly available model

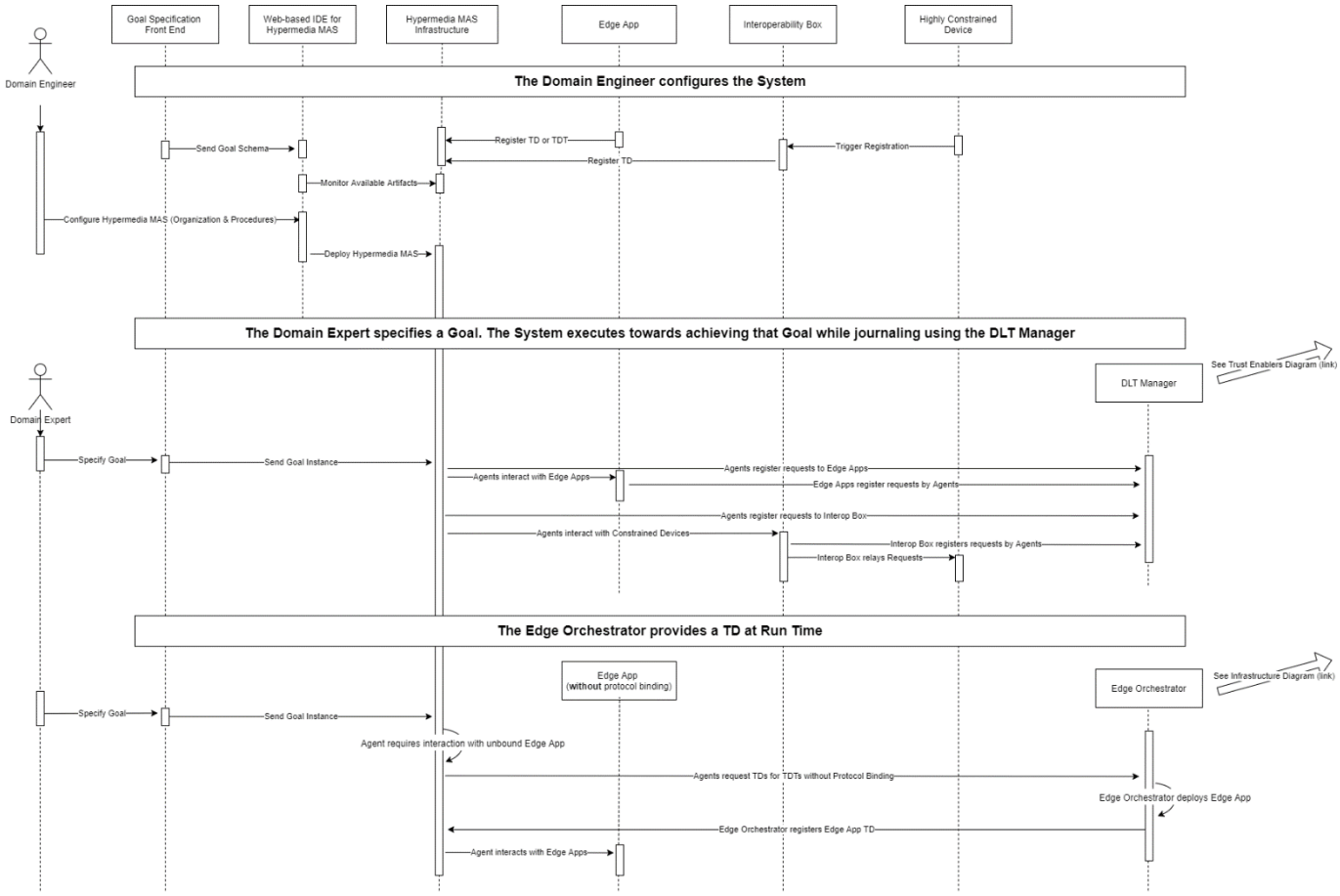

Figure 22. Interaction between Collaborative IoT enablers (sequence diagram)

Configuration: The central component in this process is the Web-based IDE for Hypermedia MAS that monitors available artifacts (and their TDs) through the Hypermedia MAS Infrastructure and is configured by the Domain Expert by means of an end-user programming approach (concretely: a visual programming language) for specifying the agent organization and procedural knowledge of individual agents, where Goal Schemas link the configured agent organization to run-time goals. To this end, Edge Apps as well as Highly Constrained Devices register with the Hypermedia MAS Infrastructure (in the latter case through the Interoperability Box) which enables the Web-based IDE for Hypermedia MAS to make their W3C WoT Interaction Affordances available to the Domain Expert as agent programming abstractions. The configured agent system is deployed to the Hypermedia MAS Infrastructure where it expects new goals according to the configured goal schema.

Execution: A process triggered by a Domain Expert who enters a new goal, through the Goal Specification Frontend, to be achieved by the configured multi-agent system. Agents then interact with each other, with Edge Apps, and with Highly Constrained Devices (through the Interoperability Box) according to their procedural knowledge and agent organization. The interactions between these components are journaled through the DLT Manager.

Edge Orchestration: This process is similar to Execution and is triggered when an unbound interaction affordance (i.e., an affordance that is provided by an unbound Edge App by means of a TD Template) is required by an agent. The agent, in this case, requests a binding for a specific interaction affordance at the Edge Orchestrator which deploys the Edge App and responds with the concrete binding of this Edge Apps' interaction affordances (i.e., a TD).

Retraining: Following the principles of federated learning and averaging [3], existing AI models at agents will use local datasets, that is the data collected over time and stored in the system data repository, in new training rounds. The aggregation of model updates at a central location allows the agents to share their local knowledge. First a

request is placed to the IAKM to enquire the availability of an updated model in any other agent within the system. If an updated AI model exists, the IAKM fetches it to the agent that placed the request. In the case of the absence of an updated model, the IAKM informs it to Global AI, in which model updating takes place. These steps are illustrated in Figure 23.

Model Update: Following the local training rounds at the Local AI Components, all AI Model updates are shared with the Global AI component, which in return carries out model averaging and evaluation. If the model averaging and evaluation shows that the model has improved performance, then it can be shared with the Local AI Components by the IAKM. Especially in the healthcare use case, aspects of personalization of models at the Local AI Components will be considered. This process takes place periodically, with training moments at the Local AI Components determined by device restrictions (idle and charging) or determined by the amount of new data collected, and at the Global AI Component in a resource-aware manner determined by the number of Local AI Components that have registered their model updates. Once an improved model is available, the Global AI Component informs the availability of an updated model to the IAKM as illustrated in the bottom half of Figure 23.

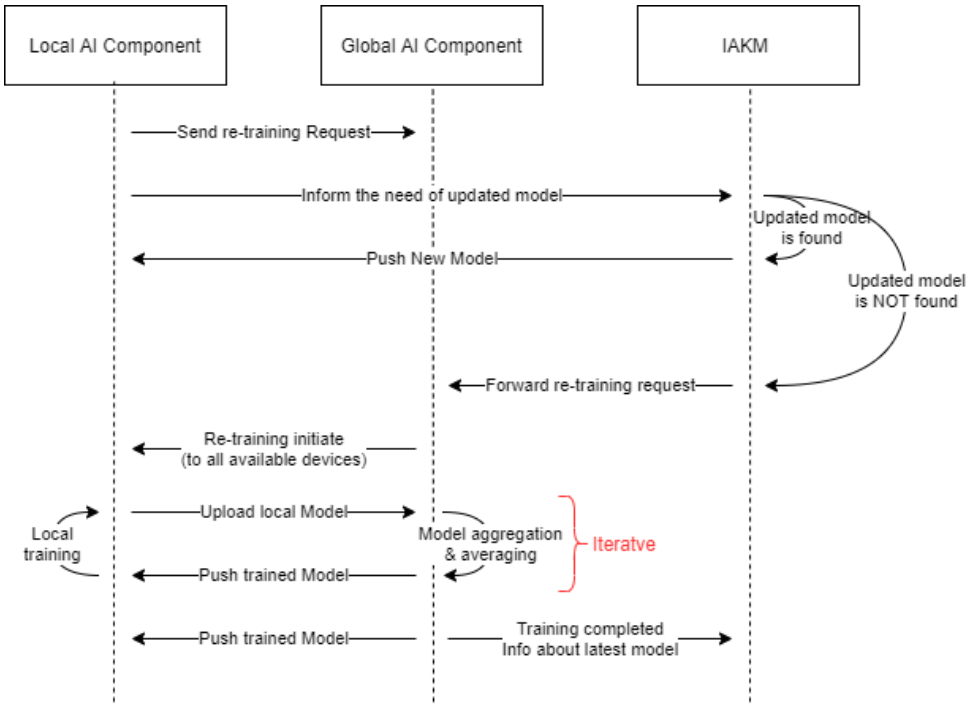

Figure 23. Interaction between AI and IAKM components

Open Call 1 contributions

The main activities that can be performed by a domain expert using the MYW Edge AI Designer are depicted in Figure 24.

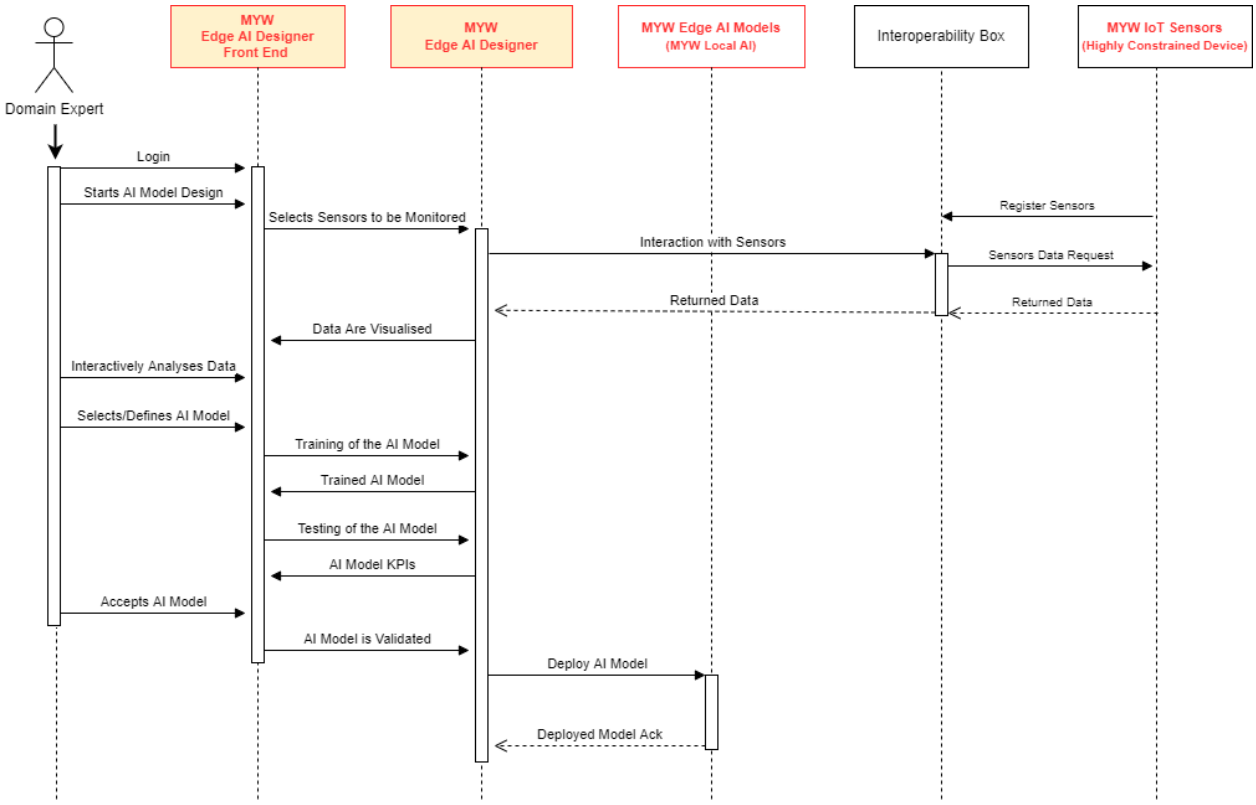

Figure 24. Process view of the MYWAI Collaborative IoT Enablers.

More specifically, the domain expert will be enabled to:

- After logging in, access and visualize the hierarchical representation of the target (e.g., tractor, robotic arm) to be monitored.
- Select and interact with relevant interoperable sensors
- Get, visualise and analyse data acquired by sensors, either real-time or historical.
- Select and label significant data subsets.
- Prepare data for AI processing (filtering, normalization, padding, windowing, etc.).
- Select the AI model to be applied for classification of data (e.g., RBF, K-nn).
- Train, test and validate the select AI model on available data.
- Deploy the trained model in the edge execution environment.

2.2.2 HUMAN-IN-THE-LOOP ENABLERS

The objective of the ***Human-in-the-Loop enablers*** is to assist the AI in solving the obstacles it might encounter while performing its functions. After storing enough human interventions at the System Data Repository, the AI will learn new behaviours for upcoming situations.

The Process diagram (Figure 25), details the steps taken - from the AI asking for help from the human until the situation is solved. The case starts with all systems in an idle state with the Local AI controlling the machine.

Rather than strongly coupling the **HIL Service** to the system, this service will be triggered through an agent that is deployed in the **Hypermedia MAS Infrastructure** and waiting for escalation requests from system components. Once the AI encounters a problem, it sends a "HelpEvent" to the Agent (which is running in the Hypermedia MAS Infrastructure according to the configuration provided by the domain engineer, see Section 2.2.1) that will propagate it to the HIL Service and eventually reaches the HIL Application. Once the event is received by the

application, the Remote Operator is alerted. The Operator will then establish the connection between the Application and the HMI (Machine to Human). After connection establishment, a "TakeOver" event would be sent from the HIL Application to the HIL Service. Where applicable, e.g., in UC3, the HIL service then invokes Edge Orchestrator, which requests communication services from TSN controller and Communication resource manager in order to establish a tactile a connection between the System and the HIL Application. Once the human accepts the "TakeOver" event, a peer-to-peer connection will be formed between the HIL application and the Client Application installed on the HMI, as well as a connection to the System, allowing a continuous transmission of an encoded (H.264) data stream. This stream is generated from the camera and metadata describing the machine's state via a combination of WebRTC and the robot control APIs (UC3) or via Zed Camera and tractor state APIs (UC1). During the stream of data and metadata, the HIL application will decode it and stream it to the HMI via the WebRTC protocol, as mentioned above. The remote Operator interacts with the HIL Application through the HMI (Human to Machine) and generates control commands that will be transmitted to the System to control its behaviour. These actions will repeat until the problem is solved.

In the Process diagram, Condition 2 illustrates the components for UC2 in which they are used mainly to support the human-in-the-loop and to collect new labelled data for the federated learning system. In UC2 and Condition 2, the output of the Local AI, which are recommendations and alerts, always requires the intervention of the human-in-the-loop, which is a physician. The physician receives a number of options through an HMI implemented as a web interface and selects one of the options. In contrast, Condition 1 supports the human-in-the-loop with HMIs using different communication protocols and requires intervention only in certain circumstances. The option selected by the physician is stored as new labelled data in the Patients Data Repository and is eventually used by the Local AI in new training rounds.

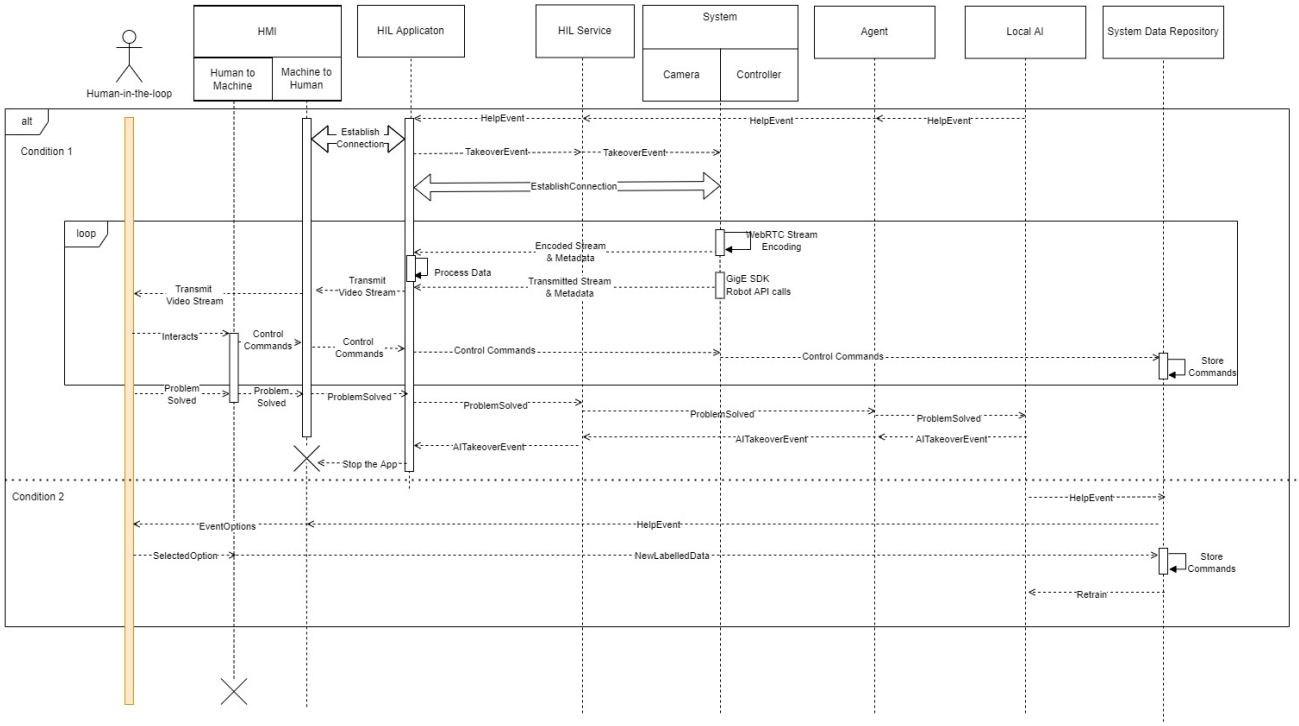

Figure 25. Process diagram of a Remote Operator intervening to help the AI solve the problem.

Open Call 1 contributions

In addition to HOLO's UC2 implementation, IKH (OC1 winner) has developed a VR application for controlling the tractor and an optional robotic arm mounted on the tractor remotely. The communication relied on a ROS bridge able to receive high level commands from the Unity application and translating them to robot-specific commands, implemented through ROS. The implementation can be seen in the photos below.

The main activities that can be performed by an operator using the **MYW Real-Time Analyzer** are depicted in Figure 26.

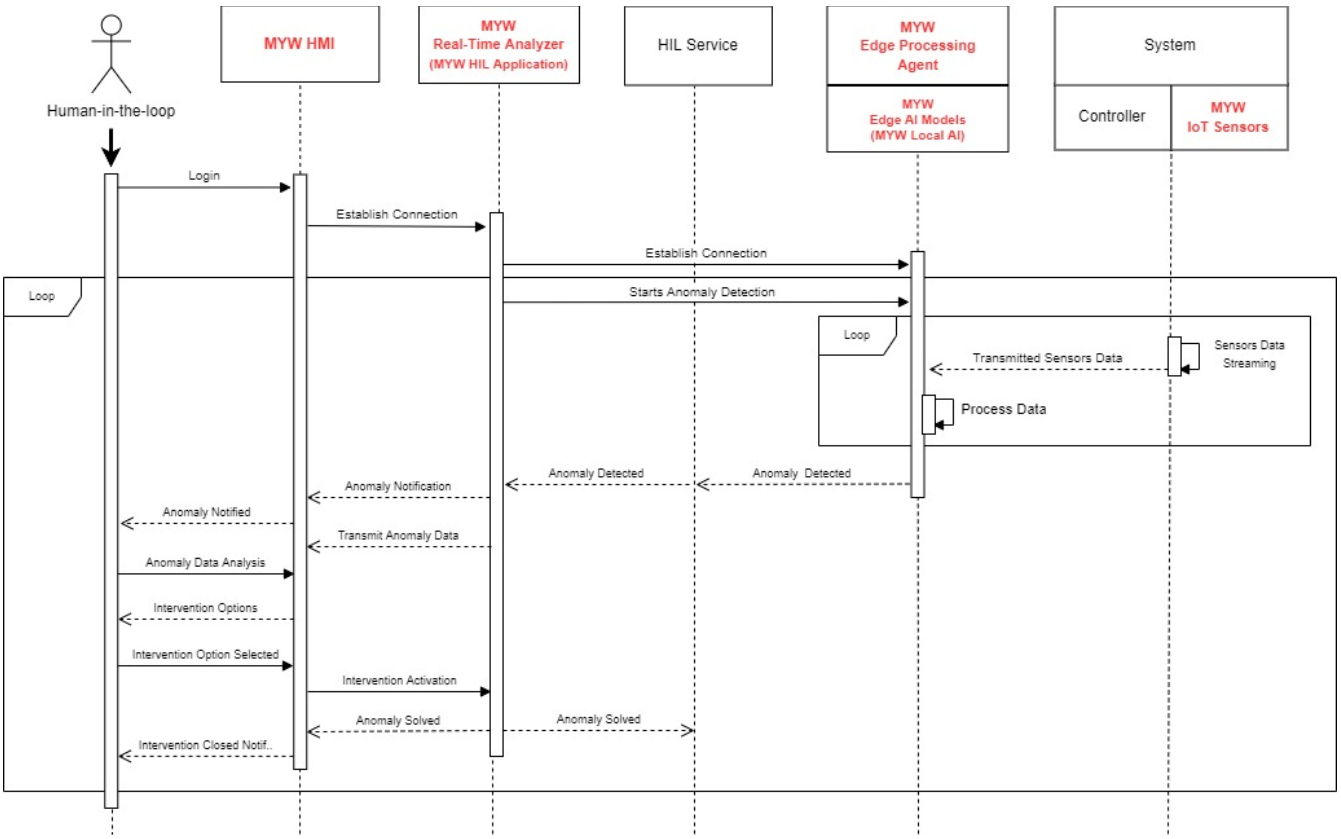

Figure 26. Process view of the MYWAI HIL Enablers.

More specifically, the operator will be enabled to:

- After logging in, connect with the target to be monitored and the related sensors.
- Local AI models are continuously running on the edge device and data from sensors are processed to detect anomalies/faults in real time.
- Once an anomaly/fault is detected the operator is notified and can start the assessment of the occurred event using the functionalities made available by the MYW human-machine interface.
- Through the interface the operator can trigger, and track intervention actions aimed at sorting out the anomalous/faulty situation.

2.2.3 TRUST ENABLERS

In the context of the IntellIoT framework, the trustworthiness components address security, privacy, and trust requirements. The generic processes covering these issues are identified and explained below.

The first process diagram (Figure 27) depicts the mitigation of an attack executed by a malicious actor. Consider a generic case of a system containing two processing nodes, namely Node A and Node B. These nodes are running their own operating system (OS), with each executing a different functionality. However, they both have the same trustworthiness components deployed on them; the Trust IDS and/or an Event Captor, the MTD Client, and the DLT Client. The Trust IDS and/or the Event Captor components continuously monitor its peers and network. When one of the two nodes is compromised, its behaviour will eventually change, resulting in a different behavioural pattern which will be detected by the IDS or captured by the Event Captor. This behaviour might include abnormal read/write system file access, abnormal network activity (e.g., data rate, packet rate), port scanning, and other such artifacts. Once the abnormal behaviour/malicious activity is detected, a warning signal is subsequently sent via the Trust Broker to the MTD Server and the Security Assurance Platform (SAP). This notification contains information about the malicious node itself and the activity associated with it.

The MTD server will then trigger the attack mitigation process, while the SAP will send evidence of the security posture change to the DLT Manager for it to be recorded in the ledger. This record (posture change log) contains identification information for the compromised node, the activity associated with it that was identified as malicious, as well as the mitigation action triggered by the MTDs (as soon as the latter information becomes available). Per the DLT protocol, in order for the log to be recorded, the transaction needs to be verified on both the DLT Manager and also transmitted to the DLT clients deployed on all available nodes (except for the malicious one) for endorsement.

In parallel, once the MTD Server receives the trigger event, it generates a mitigation configuration which contains the necessary information for the components to isolate the malicious node from the system. This configuration is subsequently sent to nodes with an MTD client deployed on them (in this case, node B), or when available the TSN controller. The MTD client running on each node removes the node identified as malicious from its own internal configuration, in order to stop accepting traffic from that node and switches to the new network configuration provided by the MTD Server. The new configuration never reaches the misbehaving node. When the TSN controller is available, it can also be used to isolate the node by excluding its endpoint from any communication. Therefore, the TSN controller reconfigures the network node associated to the malicious node. This can include removing all VLANs, closing all TAS gates and setting committed information rate to zero. This combination completely removes the compromised node from the system, effectively locking out the malicious actor and preventing further damage. All the above actions, from detection to mitigation, are further recorded in the DLT (via the SAP, which continuously monitors relevant messages on the Trust Broker), for audit purposes.

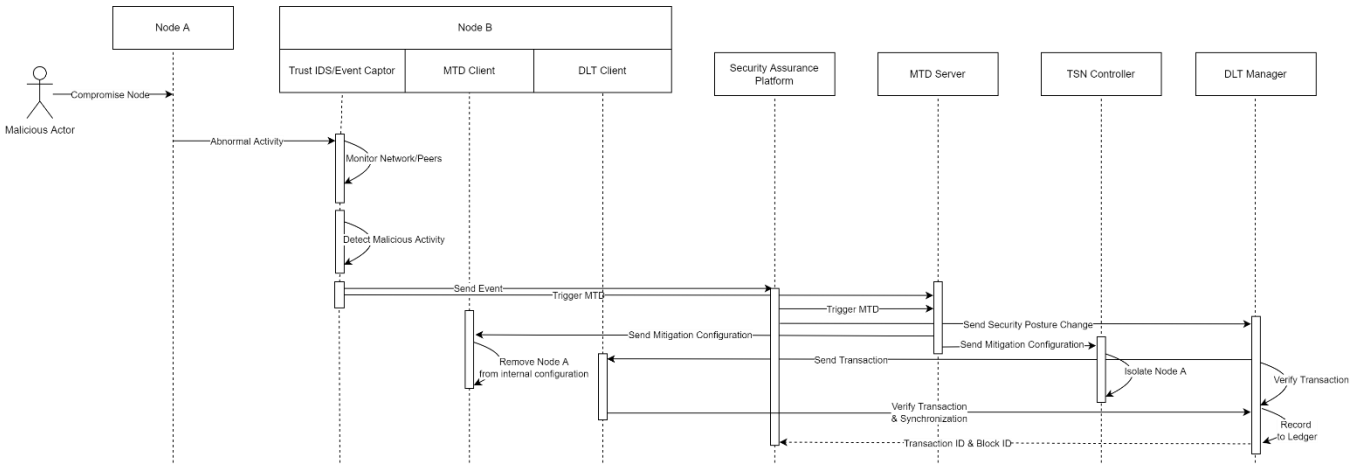

Figure 27. Mitigation of an attack execute by a malicious actor.

The second process diagram (Figure 28, left) depicts the audit process, in which the Security Assurance Platform provides an evidence-based, certifiable view of the security posture of the system, through its real-time, continuous assessment of the security module. The sequence diagram itself is straightforward; an auditor may request compliance evidence through the Security Assurance Platform's GUI. The SAP aggregates the relevant evidence through its purpose-built Monitoring tool and Event Captors, which aggregate the required evidence from multiple sources related to the operation of individual components and the overarching processes where these components are involved. In addition to the events aggregated by the SAP, additional information is requested from the DLT Manager. All this evidence is combined and provided to the Auditor, through the SAP's frontend. The report can also include Security KPIs, covering the Confidentiality – Integrity – Availability properties.

The next process diagram (Figure 28, right) depicts the request of a trust report concerning the bill of processes (BoP), progress, or proof of cost (PoC) regarding a specific job that has been requested by a customer in the context of the IntellIoT framework. This manifests in the form of smart contracts, which are generated by the human administrator with rules and agreements for the participants (machines and robots). The controls, execution, and transactions are trackable and irreversible in the ledger. When a customer makes a request, be it the BoP, PoC, or a task's progress, the ledger can provide the required data and warranty their integrity through consensus.

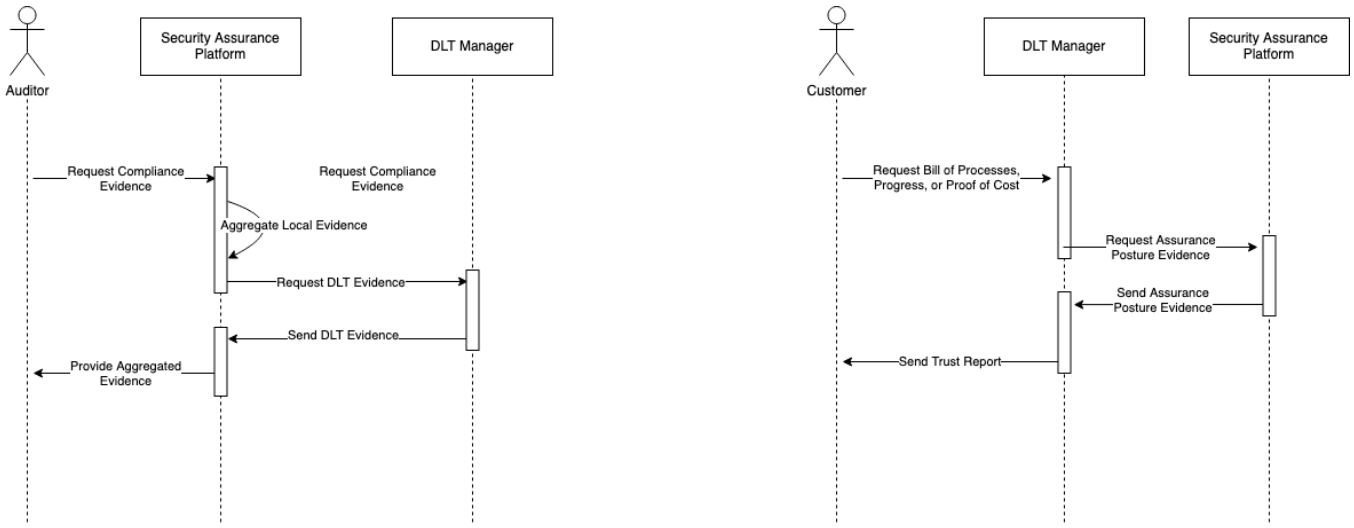

Figure 28. Sequence diagrams depicting the processes of an auditor requesting compliance evidence (left), and a customer requesting a bill of processes, progress, or proof of cost for an operation (right).

The final process diagram (Figure 29) depicts the interactions of the AAA component, when an external user or agent initiates a connection towards an internal service. Internal services have no concept of authentication/authorization to simplify their design. This is handled by an intermediate reverse proxy. The diagram depicts the process of authentication for an external user with no prior interactions. A redirection to the AAA Web UI is used in order to validate the user credentials. If the user is legitimate, a JWT token is issued and passed to the user. This can now be used at an HTTPS request header. The reverse proxy validates it and grants access by forwarding the request to the appropriate internal service. For an agent automated access, the process is slightly different. All known agents are added as trusted clients to the AAA, allowing them to get JWT tokens in an automated way and using them for their HTTPS requests.

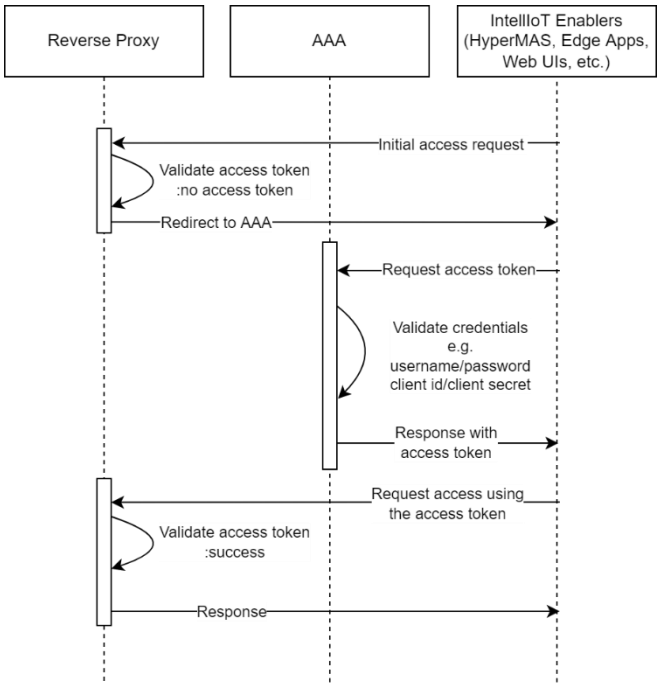

Figure 29. Sequence diagram of Authentication process

2.2.4 INFRASTRUCTURE MANAGEMENT

The components of the infrastructure management realize the dynamic management of the communication resources provided by heterogeneous networks as well as the reliable and efficient usage of computing resources of the edge infrastructure.

The sequence diagram shown in Figure 30 describes the process of deploying an Edge App in the infrastructure. This process is triggered by the system to deploy an Edge App, e.g., in case the Hypermedia MAS discovered and requires an Edge App that is not yet started/available.

Therefore, the Edge Orchestrator expects the Edge App (a pointer to or description of it) and forwards it to the Computational Resource Manager, which requests the information about the network state from the 5G LL MEC (about wireless 5G networks) and from TSN Controller (about the wired networks). The received information is merged into one network model, which is taken as input by the Computational Resource Manager to the computation of the optimal allocation of the Edge App in the edge infrastructure. The Computation Resource Manager continuously monitors the network state and updates the Edge App allocation accordingly.

The information about the optimal allocation is returned to the Edge Orchestrator, which will next prepare the involved networks for the deployment of the new Edge App, by reserving network resources through the Communication Resource Manager (for the 5G network) and the TSN controller.

Once the network resources are reserved, the Edge Orchestrator tasks the Edge Manager to deploy the Edge App and the Edge Runtime hosted by the relevant edge device is triggered to execute the Edge App.

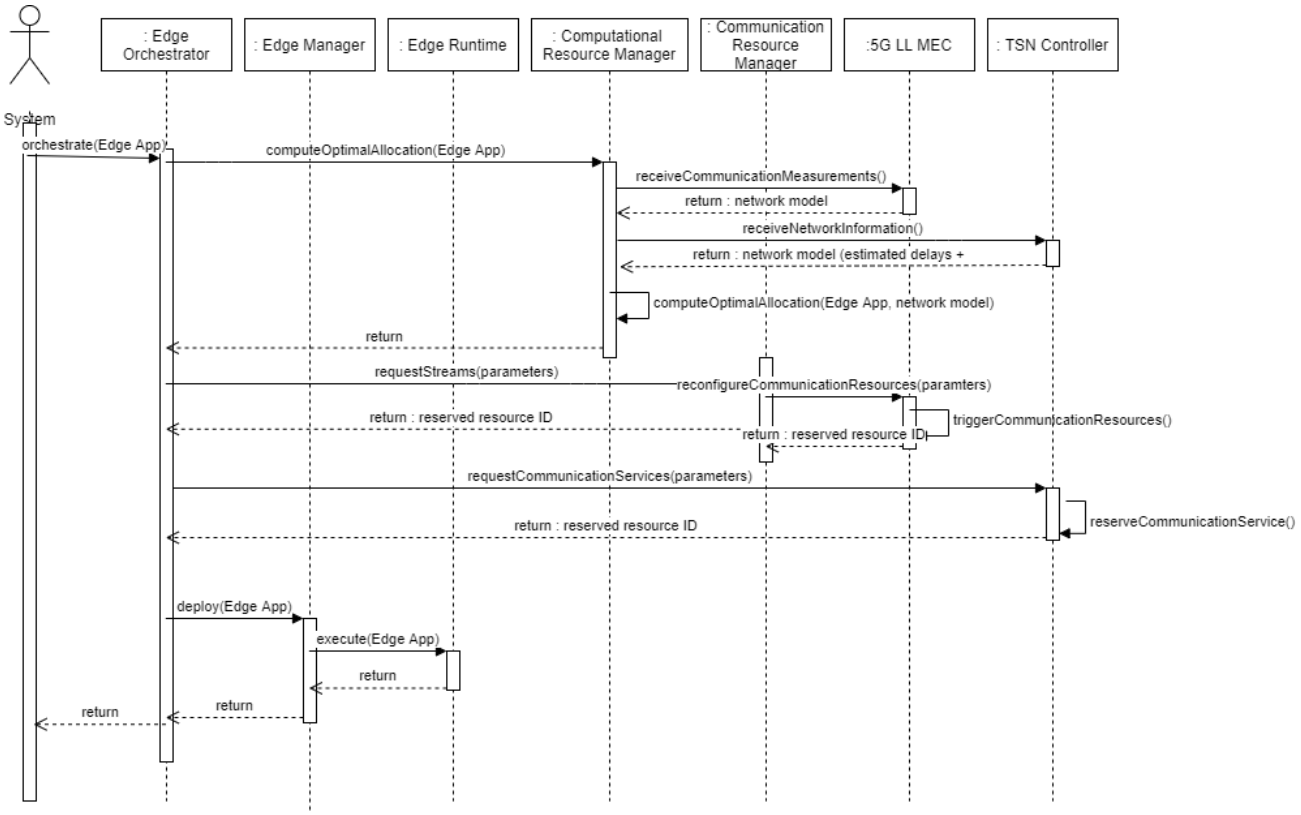

Figure 30. Sequence diagram depicting the deployment of an Edge App.

The sequence diagram shown in Figure 31 describes the process of establishing communication through the infrastructure between a human operator and a system controller (e.g., robot or tractor system) to take over control.

This process is starting with the autonomous operation of the system controller, which involves the usage of an AI. In case the confidence level of the AI is not high enough, the system controller escalates the request to an agent of the Hypermedia MAS. This agent then requests at the HIL Service that an operator takes over the control, and thereby passes on the pointer to the relevant camera, which shows the situation to be controlled, as well as skills needed for the control task. The HIL Service then forwards a Help Event to a group of HIL Applications, each of which is used by a human operator. As soon as one of the operators accepts the takeover through his/her HIL Application, the pointer to the operator as well as its ID and IP address are returned to the HIL Service. Then, the HIL Service establishes the connection between the operator and the camera by updating the Application Demand vector at Edge Orchestrator. The Edge Orchestrator requests accordingly the communication streams and services from the Communication Resource Manager for 5G networks and the TSN controller for the wired networks. It further matches the number of running spare instance against the availability constraints for the HIL Service. If required another instance will be started to reduce the takeover delay once a new service incident happens.

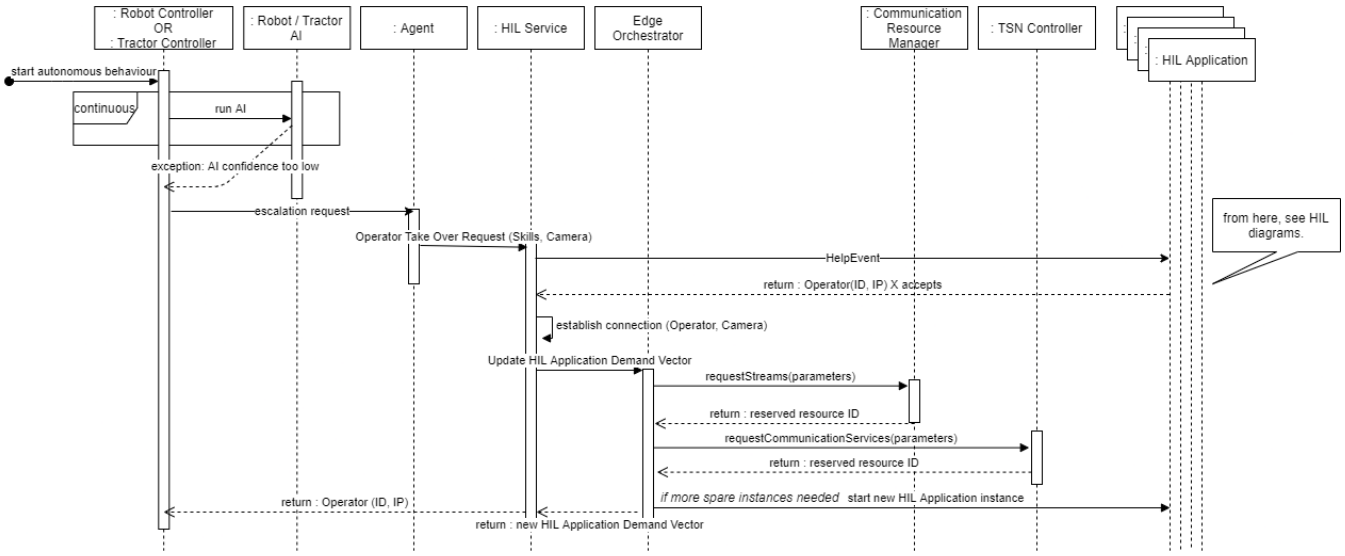

Figure 31. Sequence diagram depicting the establishing of a connection between human operator and a system controller/camera.

2.3 Development View

The Development View *(how the system is built)* per IntellIoT component group is provided in the subsections that follow.

2.3.1 COLLABORATIVE IOT ENABLERS

From a development perspective (see Figure 32 and Figure 33), the **Hypermedia MAS Infrastructure**, the **Web-based IDE for Hypermedia MAS** and the **Goal Specification Frontend** are responsible for the deployment and running of the (use-case specific) IntellIoT multi-agent systems, its configuration with procedural and organizational knowledge, and its tasking, respectively.

The **Interoperability Box** is a component that provides services and W3C WoT Thing Description (TD) compliant representations of highly constrained devices such as BLE sensors etc. For each device available to the Interoperability Box, a service that manages that device is spawned, as well as a converter service that converts data from the device's format to a format compatible with the multi-agent system. The converter service also provides endpoints and actions for each device, as described in the device's TD. Each Interoperability Box instance is responsible for registering the devices available to it as TDs to the multi-agent system, as well as providing access logs to the **DLT Manager**.

Edge Apps (External Component): The edge infrastructure provides computing capacity to offload services, which are too complex to provide them on the local machine. An edge app is the entity that bundles such a service. The edge orchestrator distributes edge apps on edge devices. It reallocates edge apps dynamically on service degradation or failures.

DLT Manager (External Component): The DLT manager provides a trusted distributed ledger for logging and tracing the activities and operations of services, for example, applications running on edge apps, collected data from high constrained devices and the Hypermedia MAS. The recorded data is first verified via a process of DLT and synchronized to the network through a consensus process. Besides, DLT is responsible for the digital identity of each actor in the systems which improve the security and guarantee the trustworthiness of data source.

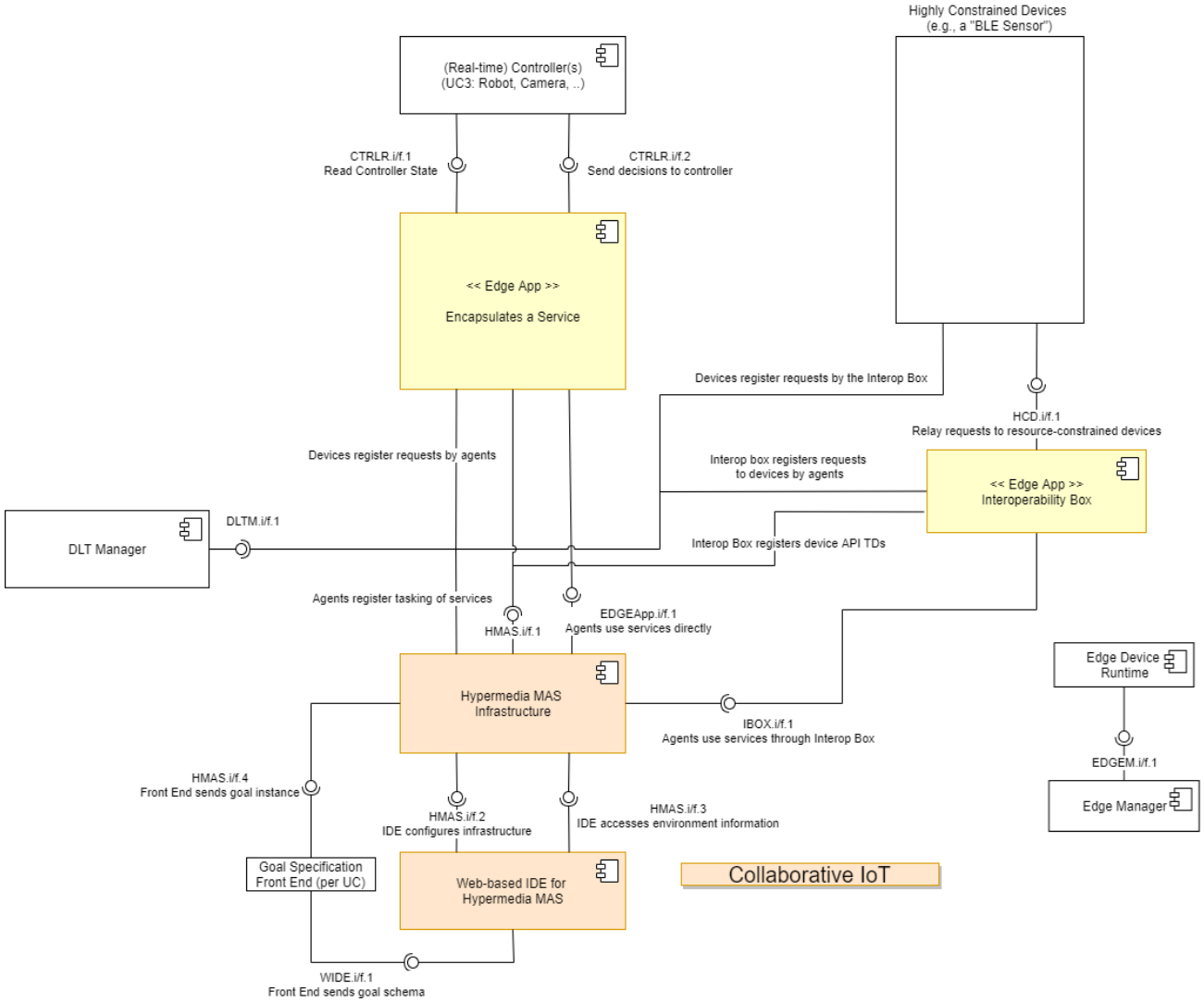

Figure 32. Development view of system-wide components, responsible for the high-level management of edge apps.

The **Infrastructure-assisted Knowledge Management (IAKM) Server** is a multi-microservice component, including an authentication module, a subscription module, a database module and a knowledge broker module. The authentication module exposes HTTPS mechanisms to authenticate the IAKM Server (edge or cloud) to other edge apps as well as to restrict access to the IAKM server services to only authenticated IAKM clients. The subscription module provides service request primitives to acquire or share particular ML-obtained knowledge. The database module is based on a non-relational database management engine (potentially extensible to relational versions), which will store ML models and datasets according to JSON defined semantics. Finally, the knowledge broker module is a secured MQTT broker providing MQTT primitives to publish or request ML models. Several instances of an IAKM server can be instantiated (e.g., to provide low latency MEC services). The IAKM provides primitives to synchronize its database module between the various instances.

The **IAKM Client** is responsible to authenticate the edge device to the IAKM Server. It exposes REST APIs for edge apps to subscribe to IAKM services. The IAKM client can be deployed as an edge app.

The **Global AI Component** connects with all the local devices with Local AI components to assist the training over the system. Here, local devices are admitted for the training upon their requests via socket connections allowing them to upload their Local AI models to the Global AI Component. Periodically, the aggregated Local AI models are averaged to produce a Global AI model, which is then shared with the connected devices until the end of training.

The **Local AI Component** is a software component that lies in the local device and has two responsibilities. 1) Retrain the Local AI model over local training data and update the models by connecting with the Global AI Component. Simultaneously, it reports communication and computation status at the local device allowing Global AI Component to schedule devices during the Federated Learning over resource-constrained networks. 2) Access the sensory data through system data repository and produce the control decisions to the controller, i.e., inference. Under the limited computing/processing capabilities at the local device, this component leverages Global AI component's processing capabilities.

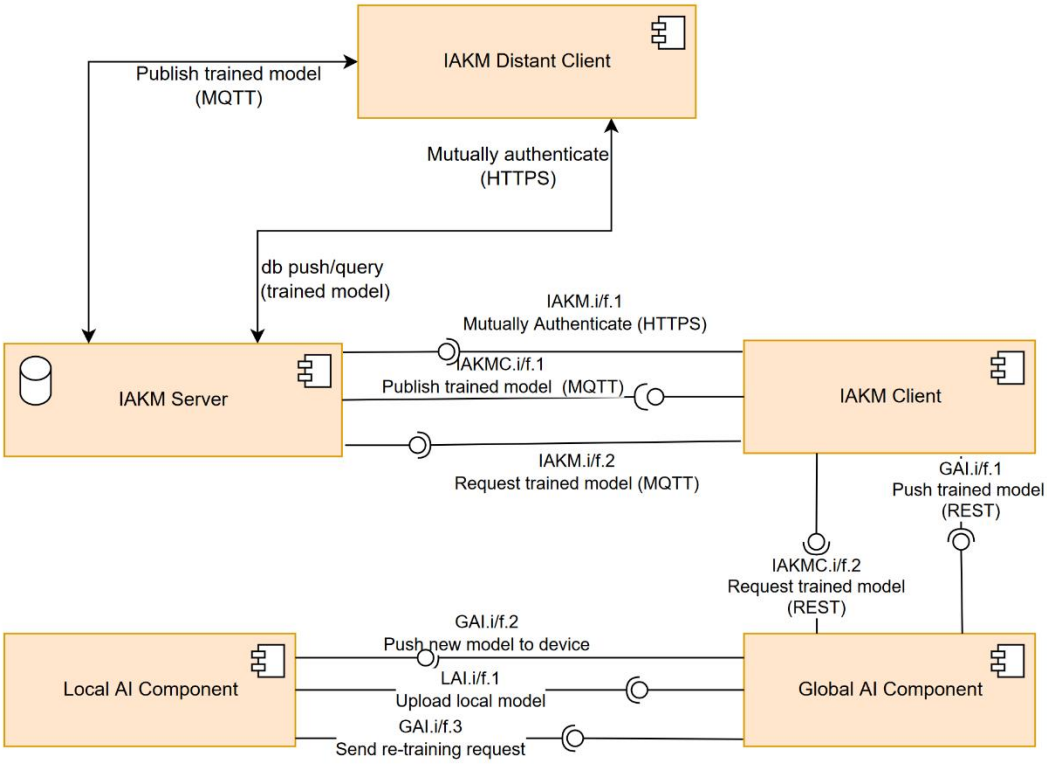

Figure 33. Development view of individually deployed components.

The above components, their key interfaces, brief description, and link to the implementation WP/Tasks of the project are presented in Table 1.

Table 1. Components of the Collaborative IoT pillar

Component	Interface	Description	Developed in
Hypermedia MAS Infrastructure	Service Usage (via EDGEApp.i/f.1)	Agents in the Hypermedia MAS Infrastructure use services provided by Edge Apps and advertised via W3C WoT TDs	T3.1
	Service Usage (via IBOX.i/f.1)	Agents in the Hypermedia MAS Infrastructure use Highly Constrained Devices through the Interoperability Box	

	Service Binding Request (via EDGEO.i/f.1 in Section 2.3.4)	Agents in the Hypermedia MAS Infrastructure request service bindings at the Edge Orchestrator (given a W3C WoT TD Template)	
	Journaling (via DLTM.i/f.2 in Section 2.3.3)	Agents, Edge Apps, and the Interoperability Box register the tasking of apps / requests by agents with the DLT Manager.	
Web-based IDE for Hypermedia MAS	Monitoring (via HMAS.i/f.3)	The Web-based IDE monitors the Hypermedia MAS Infrastructure for currently available interaction affordances (i.e., TDs and TD Templates)	T3.1
	Configuration (via HMAS.i/f.2)	The Web-based IDE configures the multi-agent system in the Hypermedia MAS Infrastructure with procedural and organizational knowledge as well as a goal schema.	
Goal Specification Frontend	Send Goal Instance (via HMAS.i/f.4) and Goal Schema (via WIDE.i/f.1)	After the Domain Expert has configured a new goal, this goal is transmitted to the (configured and running) multi-agent system in the Hypermedia MAS Infrastructure.	T3.1
Infrastructure-assisted Knowledge Management (IAKM) Server	Authentication (IAKM.i/f.1)	The IAKM server supports authentication mechanisms for IAKM clients or other IAKM servers over HTTP with TLS	T3.2/ T4.3 / T4.4
	Service usage (IAKM.i/f.2)	The IAKM server provides knowledge brokership to other IAKM servers or clients over MQTT with TLS	T3.2/T4.3
Infrastructure-assisted Knowledge Management (IAKM) Client	Service usage (IAKMC.i/f.1)	The IAKM client sends knowledge requests or receives knowledge data over MQTT using TLS (MQTTS)	T3.2/T4.3
	Request trained model (IAKMC.i/f.2)	The IAKM client may request a trained model from the Local AI component.	T3.2/T4.3

Global AI Component	Push trained model (GAI.i/f.1)	Once retraining is completed, the AI model is shared with IAKM client.	T3.2/T3.2
	Push new model to the device (GAI.i/f.2)	Local models are received from the Local AI Components, which in return aggregated and averaged to generate a global model. Then, the global model is shared back with the Local AI Components.	T3.2
	Retraining request (GAI.i/f.3)	Based on a retraining request obtained from IAKMC.i/f.2, a retraining sequence is initiated by informing the Local AI components.	T3.2
Local AI Component	Upload local model (LAI.i/f.1)	During retraining, at each iteration, the trained local AI model is shared with the Global AI Component.	T3.2/T4.3
	Retraining request (GAI.i/f.3)	With failures of the inference, this component generates and sends a retraining request to the Global AI Component.	T3.2/T4.3

Open Call 1 contributions

From a development perspective, the **MYW Edge AI Designer** through the corresponding front end is in charge of supporting a domain expert through a set of services enabling the access to structured and standardized information about a target and the related sensors, with the aim of facilitating the analysis of data and the definition of AI anomaly/fault models to be then locally deployed and executed at the edge/very edge for monitoring/maintenance purposes. Figure 34 depicts the MYW Collaborative IoT Enablers and their interfaces.

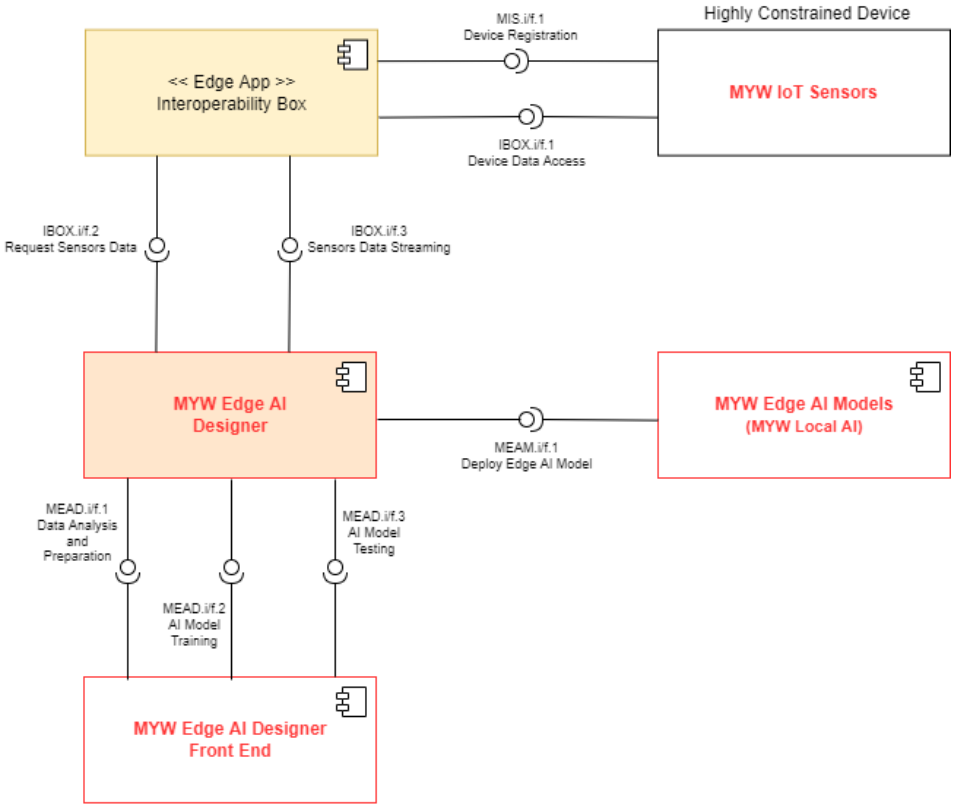

Figure 34. Development view of the MYWAI Collaborative IoT Enablers.

At a high-level, brief descriptions of the above represented MYW and IKH components and their key interfaces are summarised in Table 2.

Table 2. MYW and IKH Collaborative IoT components & their interfaces

Component	Interface	Description	Developed In
MYW Edge AI Designer	Data Analysis and Preparation (MYWEAD.i/f.1)	Access to data related to selected targets/sensors, for visualisation, analysis, preparation, and labelling activities performed by the domain expert through the MYW EAD Front End	OC#1
	AI Model Training (MYWEAD.i/f.2)	Design, development, and training of anomaly AI models functionalities, available to the domain expert through the MYW EAD Front End,	

	AI Model Testing (MYWEAD.i/f.3)	Testing and validation of anomaly AI models functionalities, available to the domain expert through the MYW EAD Front End,	
MYW Edge AI Models	Deploy Edge AI Model (MYWEAM.i/f.3)	Deployment of validated anomaly AI models to the MYW Edge Processing Unit execution environment as Docker containers.	OC#1
MYW IoT Sensors	Device Registration (MYWIS.i/f.1)	Interoperable and standardised sensors data access	OC#1

2.3.2 HUMAN-IN-THE-LOOP ENABLERS

As mentioned above, the Human-in-the-Loop scenario is triggered as a response to an event. Hence, apart from the HIL Service and HIL Application, all other components must fulfil a function even before the event gets triggered. The software components depicted (Figure 35) and mentioned below address either HIL enabler specific aspects of components, or the development of the HIL Service and Application.

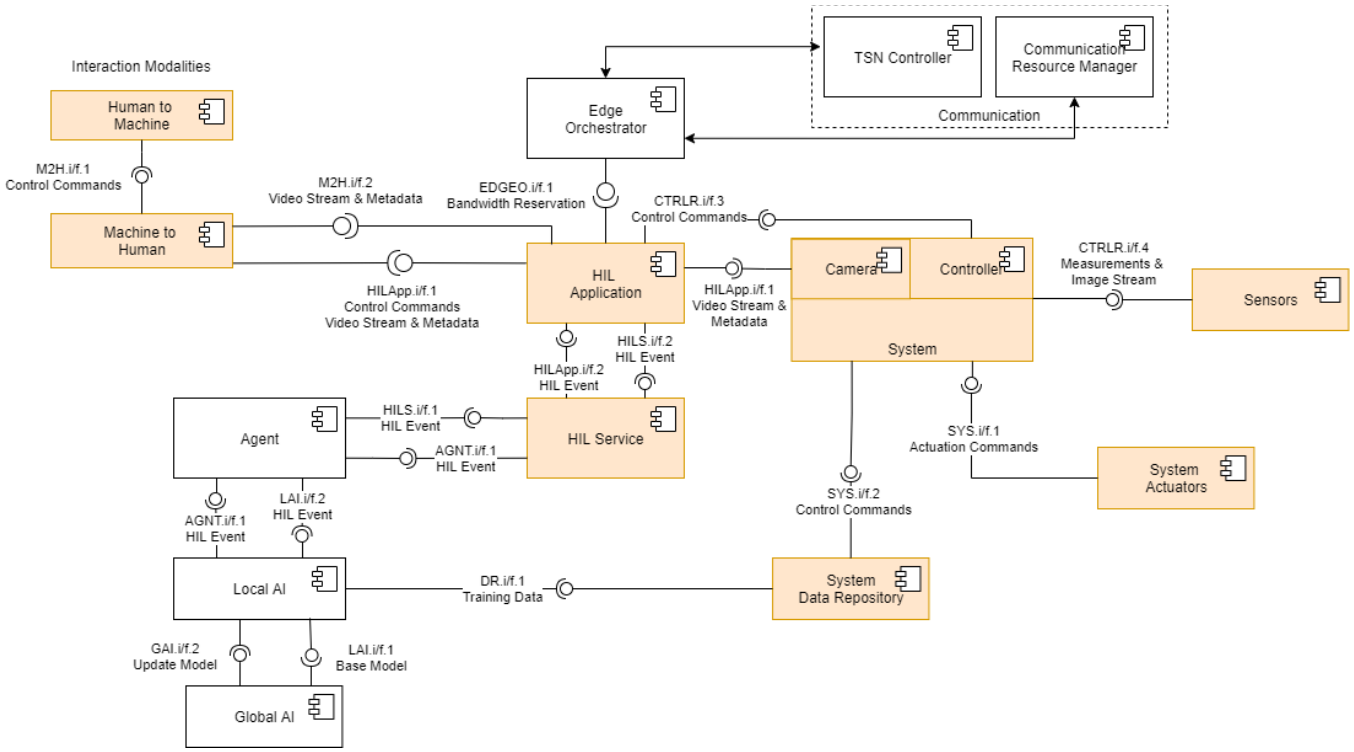

Figure 35. Development diagram of the HIL enablers.

The key components to enable the Human-in-the-Loop are the **HMIs**, which are divided into two parts:

- **Humans to Machine:** They allow the human to interact with the virtual environment to send control commands to the HIL Application. They are the Stylus for the AR Application & the VR Controllers for the VR Application. Though the respective headsets also provide input options, such as tilting the head or gaze, they are not included in this definition, to avoid confusion with the next one.

- **Machine to Human:** They are the HoloLens 2 for the AR Application & the Oculus Quest 2 for the VR Application. They are responsible for displaying the virtual space containing all UI elements for the human to interact with.

The logic and app architecture behind the interaction options are developed in Unity3D and form the **HIL Application**. It is a collection of various interfaces (Zed Camera SDK, Robot control and tractor state REST APIs, WebRTC, Edge Orchestrator APIs, HIL Service APIs, etc.), and UI elements that hook up to the interfaces and is controlled by a clear app logic to enable a human take-over. The app receives and displays the video stream from a camera installed on the machine, useful metadata within a dashboard and potentially also further relevant data, such as weather alerts. The entire application, including the various data streams are then send through WebRTC to the HMIs for the human to view. Furthermore, the Human Operator can input control commands, which are transmitted via the HIL Application.

The controllers receive their information from the **Sensors**. Sensors in UC3 are implemented as position sensors in the robot, which are accessed through the UR5 API, which provides complete information about position and movement of the robot. Additionally, a GigE vision camera is used to provide workpiece images to AI and to provide a video feed to a human operator.

The **System Actuator** is realized in the form of a UR5 robot[3], comprising the UR5 controller.

The **HIL service** receives help requests from an AI. The HIL Service is triggered through an escalation by the AI component of the robot or tractor that is brokered by an agent in the **Hypermedia MAS**. That agent is programmed to handle the escalation by the user of the Web-based IDE for Hypermedia MAS, which requires access to the TD of the HIL service. It maintains knowledge about availability, reachability, and relevant properties of available **HIL Applications** with the help of the **Edge Orchestrator**. In case of a help request from a system, it contacts one or more available HIL applications via known addresses. These can be pre-configured or registered at runtime. It establishes a connection to the HIL application that accepted its request. Where stringent requirements on timing apply, it requests communication services via **Edge Orchestrator**. These requests are handled via REST requests.

As mentioned earlier, the **System Data Repository** will store all human interventions and data that is needed from the sensors to support the human-in-the-loop. In use cases 1 and 3, the System Data Repository is used to store the local sensory information at the device (tractor and robot); these data will be used for local AI model training as well as inference. In use case 2, the Data Repository will store only patient data that is required for making the reports and the recommendations that need to be sent to the physicians.

The **TSN controller** is implemented as an edge app, receives communication service requests from the **Edge Orchestrator** via its REST-based northbound interface, and configures network nodes, so that tactile interaction between a human operator and the system is possible and reliable. The southbound interface for device configuration is preferably netconf, ideally using standardized YANG models. To integrate legacy devices, southbound interface adapters, e.g., for proprietary CLI interfaces, might be used.

The **Communication Resource Manager** is a multi-microservice platform including a 5G resource manager, and 5G RAN controller microservices. They can be deployed as multi-docker containers via a Docker Compose YAML file, or to Kubernetes or OpenShift. Snap deployments are also investigated. The 5G resource manager southbound interface is a REST technology, whereas its northbound as well as the controllers' northbound interface connects this component with the Edge Orchestrator. Finally, the RAN intelligent controller southbound interface supports ORAN FlexRIC APIs, whereas the NVF controller southbound interface supports both NVF APIs as well as Virtual Infrastructure Management (VIM) APIs.

The decisions of the human-in-the-loop during the interventions are recorded as new labelled data. To leverage this data, the **Local AI** Component uses it during the retraining of its local AI model. This is followed by the federated learning process as described previously in Section 2.1.1.

The above components, their key interfaces, brief description, and link to the implementation WP/Tasks of the project are presented in Table 3.

[3] https://www.universal-robots.com/products/ur5-robot/

Table 3. Components in the human-in-the-loop pillar

Component	Interface	Description	Developed in
Human to Machine	M2H.i/f.1	Displays the stream received from the HIL Application to the human.	T3.3
Machine to Human	HILApp.i/f.1	Passes on coordinates and controls to the HIL Application	T3.3
5G Communication Resource Manager	5G	Provides 5G communication between edge apps and the infrastructure	T4.2
	5G resource request/triggering	REST requests enable the Edge orchestrator to trigger the creation of 5G resources	T4.2
	Monitoring	Provides 5G resource monitoring	T4.2
System	CTRLR.i/f.3	WebRTC or REST APIs	UC components
	HILApp.i/f.1	WebRTC or GigE APIs	UC components
System Data Repository	SYS.i/f.2	Store local sensory data to be used for local AI model training.	T3.2, T3.3
HIL Application	M2H.i/f.2	XR Application allowing the user to control the machine remotely while rendering the machine's environment in a virtual space.	T3.3
Controller	CTRLR.i/f.3	Typically REST API. Tractor controller (UC1), Smart phone (UC2), Robot controller (UC3)	T5.1 / T5.2
Sensors	CTRLR.i/f.4	Tractor (UC1), Smart phone and wearable / medical sensors (UC2), Robot sensors (UC3)	T5.1 / T5.2

System Actuators	SYS.i/f.1	Tractor actuators (UC1), Physician interventions (UC2), Robot actuators (UC3)	T5.1 / T5.2
HIL Service	DR.i/f.1	HIL Service is called on help requests by the AI.	T4.1

Open Call 1 contributions

From a development perspective the **MYW Real-Time Analyser**, i.e., the MYWAI HIL application, through the dedicated human-machine interface is in charge of supporting a human operator in the loop by notifying behavioural anomalies/faults affecting a system and detected through the AI-based processing of real time data acquired by sensors and executed at the edge of the monitored target. Figure 36 depicts the MYW HIL Enablers and their interfaces.

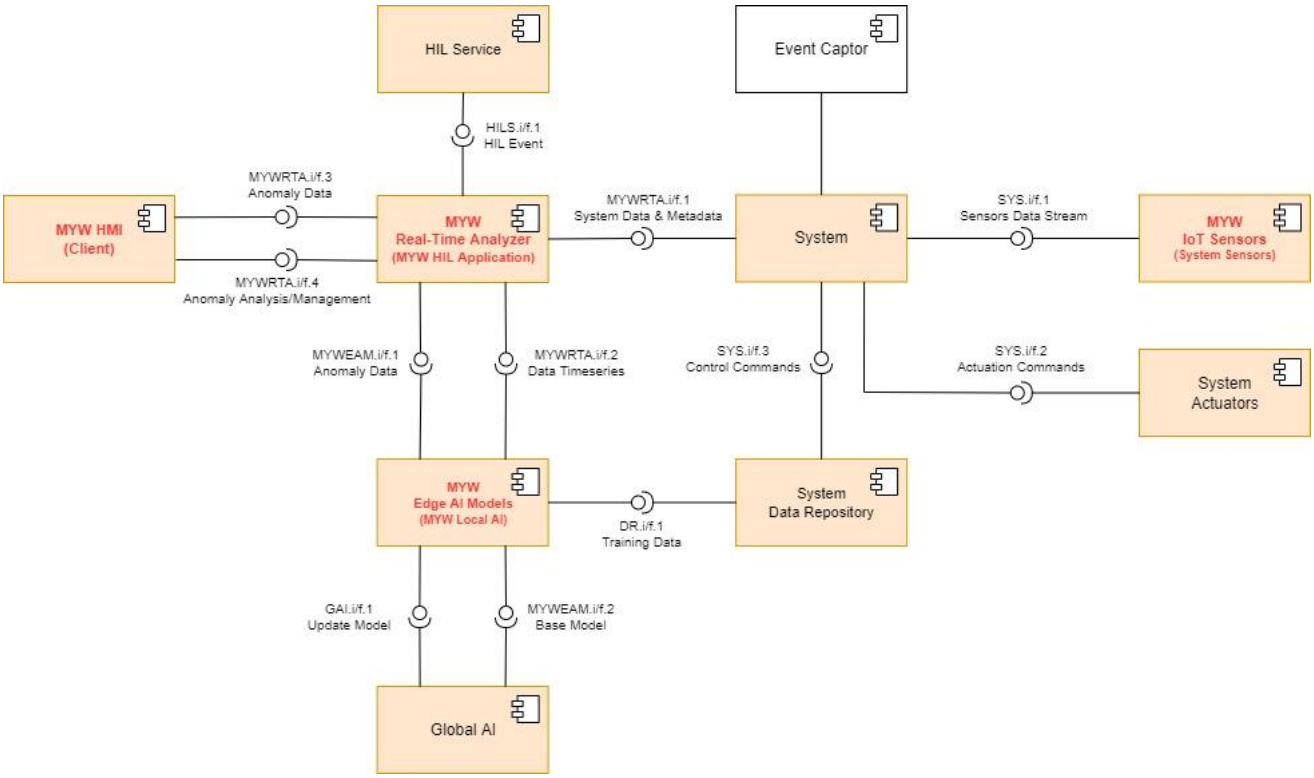

Figure 36. Development view of the MYWAI HIL Enablers

High-level brief descriptions of the key interfaces exposed by the above represented MYW components are collected in Table 4.

Table 4. MYW HIL components & their interfaces

Component	Interface	Description	Developed In
MYW Real-Time Analyzer	System Data and Metadata (MYWRTA.i/f.1)	Gathering of data and metadata from the monitored system/sensors, for further automatic AI processing as well as HIL analysis in case of anomaly.	OC#1

	Data Timeseries (MYWRTA.i/f.2)	Data stream(s) provided to the MYW Edge Processing Unit execution environment for AI-based processing by MYW Edge AI Models.	
	Anomaly Data (MYWRTA.i/f.3)	Notification of an anomaly, with annexed basic information and metadata, to the operator (HIL) through the MYW HMI.	
	Anomaly Analysis/Management (MYWRTA.i/f.4)	Access and retrieval of anomaly data, as well as real-time equipment information and status made available to the operator through the MYW HMI.	
MYW Edge AI Models	Anomaly Data (MYWEAM.i/f.1)	Once an anomaly is automatically detected by a MYW local AI model, related contextual information is sent to the MYW RTA for notification to the operator and further actions.	OC#1
	Base Model (MYWEAM.i/f.2)	Once an anomaly has been detected and managed, retrieval of a reference base model for retraining of both local and global AI.	

2.3.3 TRUST ENABLERS

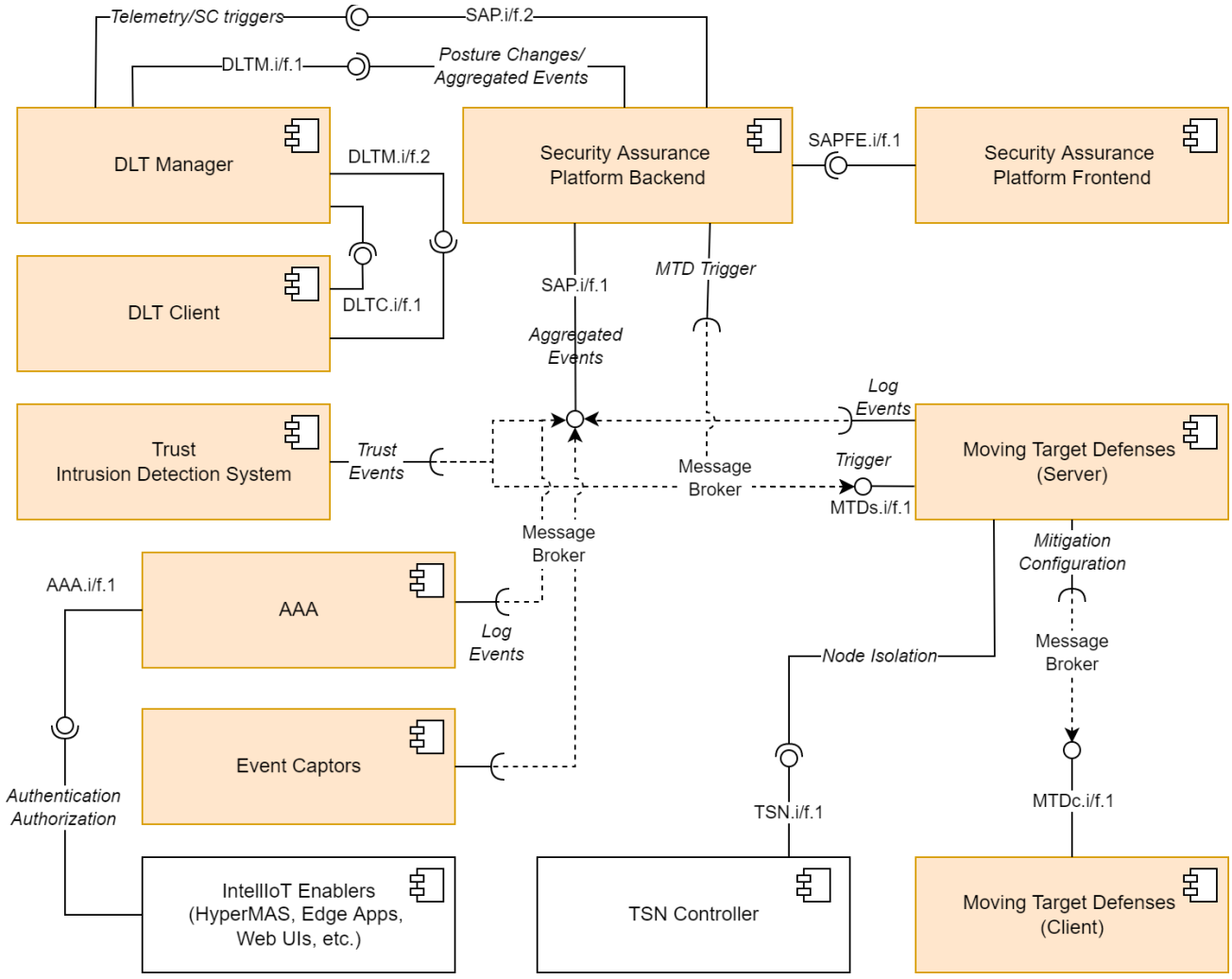

Figure 37: Trust enablers development diagram.

As stated previously, the Security Assurance Platform is central to the trustworthiness components, while a Trust Broker is used as the single integration point of all messaging between trust enablers. Overall, the interaction between trust enablers & other components is made possible via 2 ways: i) the (RabbitMQ-based[4]) Trust Broker, ii) appropriate REST APIs. These are depicted in Figure 37.

The Trust Broker implements an AMQP-based (Advanced Message Queuing Protocol[5], approved as an international standard in the form of ISO/IEC 19464) messaging bus that allows communication between the trust enablers and external components by acting as the mediator between them. It receives incoming messages from a sender (publisher) and forwards them to the recipient (consumer), thus keeping the sender and receiver completely isolated. Such interactions include:

- The Event Captors, which utilize the publish/subscribe (pub/sub) messaging protocol, an asynchronous service-to-service communication typically used in serverless and microservices architectures. These Event Captors need to be deployed separately on each device (physical or virtual) to be monitored. Their function may be initiated manually, or automatically by the SAP when a monitoring assessment is initiated via the SAP Frontend. Once an Event Captor is initiated, it begins to aggregate evidence, such as network traffic and system events and publish them. These events are subsequently copied to a queue, from which the corresponding consumer (in this case the SAP's monitoring tool) can receive them through a push/pull

[4] https://www.rabbitmq.com/
[5] https://www.amqp.org/

method. In the context of IntellIoT, different Event Captors are developed to cover all needs of the implemented use case scenarios. For example, in the Healthcare Use Case, different Event Captors are responsible for monitoring the network parameters (deployed on the network edge), trust-related operational parameters (deployed on physicians' workstations), and privacy-related parameters (deployed on all relevant hospital assets handling patient data), while in the Manufacturing Use Case Event Captors are tailored to capture telemetry from the Robotic arm.

- The Trust IDS instances that use the RabbitMQ message broker to publish their local trust values for the nodes they are communicating with along with warnings for offending nodes. These values are accessible for the SAP/MTD to consume. SAP aggregates all the information generated from the Intrusion Detection System instances in order to determine when an action needs to be taken against an offending node.
- The Moving Target Defences (MTDs). Once the SAP receives an Intrusion Warning from the IDS, it needs to trigger a mitigation mechanism, namely the MTDs. These can be divided into two distinct components: i) the server MTD, and ii) the client MTD.
    - The MTD server, which is deployed alongside the SAP, is responsible for receiving trigger signals via the RabbitMQ message broker, subsequently generating a mitigation configuration. Trigger signals can be issued either by an IDS instance or SAP. The mitigation configuration contains the necessary information to isolate the malicious or compromised nodes in the system and is transmitted to both the TSN controller (if available) via a REST API, and the client MTD via the message broker.
    - The MTD clients, which need to be deployed on each device alongside the ECs and IDS, receive the corresponding mitigation configuration from the MTD server, and update their internal routing tables to drop all traffic from the isolated node. Also, the encryption configuration is updated with different keys and/or algorithms, to mitigate packet sniffing. Finally, the tunnels could be moved to different IPs and/or ports, so that the isolated node is unable to extract information about the network topology.
- The Security Assurance Platform (SAP) itself, since the broker is not just used by the SAP for ingesting aggregated evidence, but also to trigger and execute mitigation strategies encoded within its incident response component (e.g., to execute a Playbook action that dictates triggering the MTDs).

In addition to the Trust Broker, appropriate REST APIs are used, when necessary, to enable the communication with additional components and outside interfaces. The SAP/DLT is one such API, which enables the communication of the SAP with the Distributed Ledger Technology used in the IntellIoT framework. This interplay of the SAP with the DLT serves to record all trust posture changes and aggregated, trust-related, events (e.g., sent attack mitigation triggers) to the ledger in order to ensure the aggregated data integrity.

Furthermore, although not implemented in IntellIoT, it could easily be extended to report all Smart Contract (SC) term violations to the SAP, consequently notifying the user of tampered data.

The objective of DLT nodes is to maintain the reliability of the data stored on the ledger. When a new piece of data or block is added to the ledger, the DLT node will communicate the block to other nodes on the network. Based on the validity of the new block and the type of node, full nodes can reject or accept the block after the consensus process. Once a new block is accepted by the DLT nodes, the information is stored and saved on top of the pre-existing blocks. In the scope of this project, DLT nodes are classified into two categories: DLT manager and DLT light client and also considered as lightweight or Simple Payment Verification (SPV) nodes.

DLT managers have the single copy of an entire ledger history including transactions, timestamps, and all generated blocks. For instance, a full Bitcoin or Ethereum node would host all information regarding every single transaction since the start of the network to the present moment. Meanwhile, DLT clients are usually installed in limited capacity devices and responsibility for generated transactions, then forwarded to the DLT manager for mining process. DLT light clients then can be synchronized with the DLT manager to update about the state of the ledger. When the DLT clients want to query or request the recorded data, it should consult to DLT manager.

The AAA solution employed is the open-source Identity and Access Management tool, Keycloak[6]. It provides support for authorization, by using the OAuth 2.0 protocol, and authentication, using the OpenID Connect protocol. Both are industry standard protocols with widespread adoption and support. It can be accessed indirectly through a reverse proxy for automated access of services or directly through Web UI for user logins. In both cases, the AAA.i/f.1 interface is used.

The TSN Controller offers a REST interface to isolate malicious nodes, which is used by the MTDs to trigger the TSN controller to isolate a particular node, as identified by its MAC address or node name. The TSN controller realizes the isolation by features provided by the switch the node is connected to, e.g., by removing all VLANs, permanently closing all TAS gates and setting the Committed Information Rate of PSFP to zero. The interface to the network nodes is preferably netconf, but proprietary CLI interfaces are possible.

The above components, their key interfaces, brief description, and link to the implementation WP/Tasks of the project are presented in Table 5.

Table 5. Security, Privacy and Trust enablers

Component	Interface	Description	Developed in
Security Assurance Platform Backend	SAP.i/f.1 (via Message Broker)	Receives aggregated trust-related events from Event Captors, the Trust IDS, MTDs, and other interacting entities	T4.4
	SAP.i/f.2	Receives SC term violations from the DLT Manager	
AAA	AAA.i/f.1	Allows the authentication and authorization of modules deployed on servers and devices	T4.4
Trust Intrusion Detection System	Low trust warning (via Message Broker)	Allows IDS instances deployed in different devices to forward their trust state to the MTD Server using MTDs.i/f.1	T4.4
TSN Controller	TSN.i/f.1	Isolates malicious nodes when commanded by MTD.	T4.2, T4.3
DLT Manager	DLTM.i/f.1	Receives posture changes and aggregated events from the Security Assurance Platform.	T3.4
	DLTM.i/f.2	Responsible for DLT mining process, have a full version of distributed ledger.	T3.4
DLT Client	DLTC.i/f.1	Generate and sign transactions, then forward to DLT manager. Have a simplified version of distributed ledger.	T3.4
Moving Target Defences (Server)	MTDs.i/f.1	Receives a trigger from Trust IDS or the SAP to initiate the client-side MTDs and mitigate an attack.	T4.4
Moving Target Defences (Client)	MTDc.i/f.1	Receives the generated Mitigation Configuration to apply in order to isolate malicious processing nodes.	T4.4

[6] https://www.keycloak.org/

2.3.4 INFRASTRUCTURE MANAGEMENT

Figure 38 outlines the interfaces between the components for the infrastructure management. While the components in beige are related to the management of the edge / computing side of the infrastructure, the blue components are managing the network side of the infrastructure, both the wired TSN as well as the wireless 5G networks.

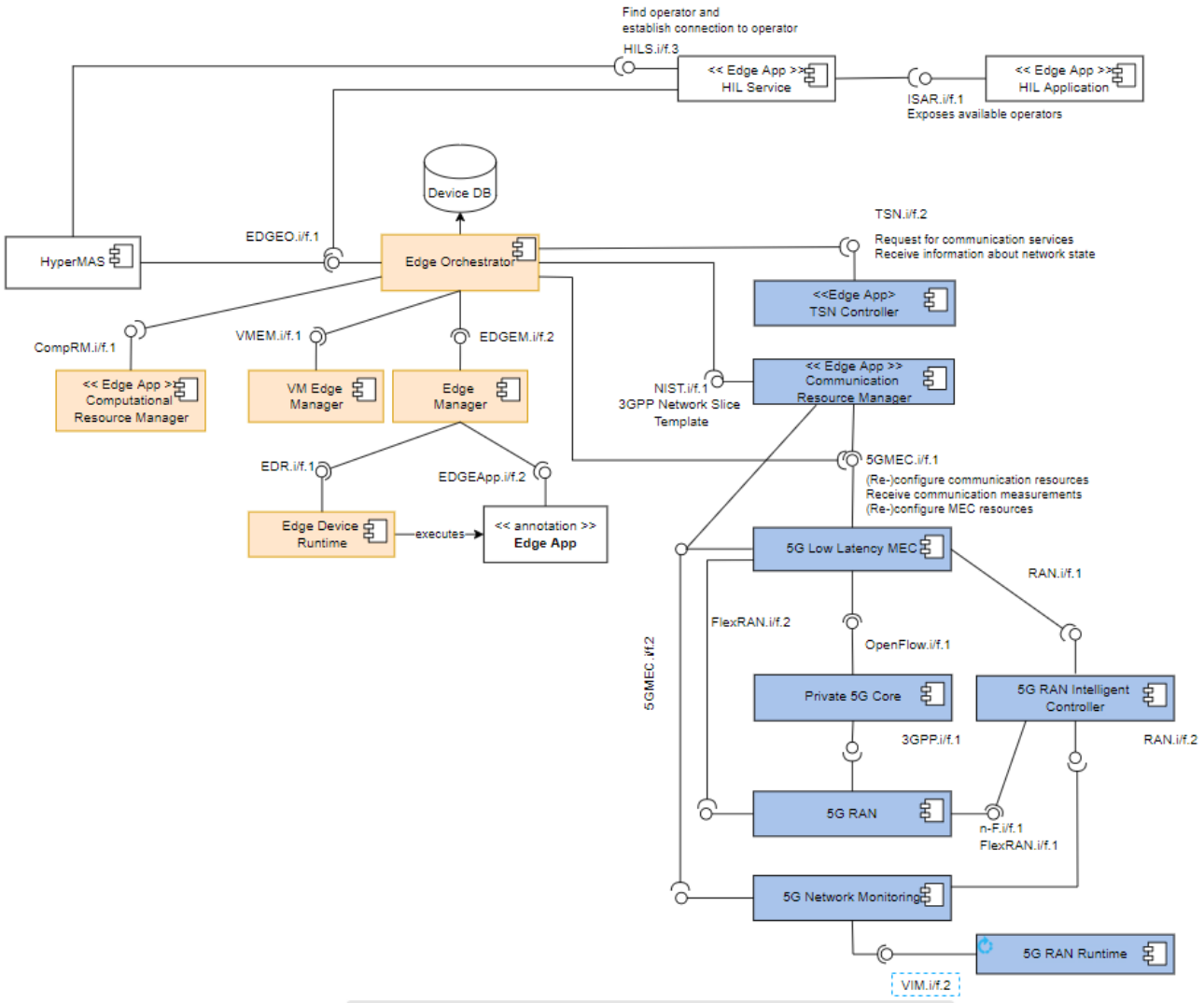

Figure 38: Infrastructure Development View.

The EDR API of the **Edge Device Runtime** is used by the **Edge Manager** for lifecycle operations of **Edge Apps** on that particular **Edge Device**. The EDGEO API of the **Edge Orchestrator** is used by the Hypermedia MAS to trigger the allocation or deallocation of resources for **Edge Apps** in the edge infrastructure. It is also used by the HIL Service to (1) establish the connection between operator and machine and (2) to notify the Edge Orchestrator that the demand of the HIL application has changed. Further it provides a service interface, which is consumed by a web UI, for provisioning and managing of **Edge Apps**. The EDGEM API of the **Edge Manager** is used by the **Edge Orchestrator** for lifecycle operations of **Edge Apps** and **Edge Devices**, which are managed by the **Edge Manager**.

The CompRM API of the **Computational Resource Manager** is called by the Edge Orchestrator and returns its result of the optimal allocation to the Edge Orchestrator. The input interface for the Computational Resource Manager needs to contain information about:

- IoT application configuration (functions) and constraints
- Edge configuration (devices' compute capacity, energy consumption, etc.)

- Network configuration (network link details on bandwidth, delay, jitter etc.)

The **5G Low Latency MEC** provides a FlexRIC/FlexCN API as an abstraction between the LL-MEC and the 5G RAN, 5G Core and 5G RAN Intelligent controller components. The LL-MEC is connected to MEC applications, exposing ETSI MEC APIs, to monitor the traffic as well as the state of the network and optimize accordingly. The LL-MEC operates as a Docker microservice.

The **Private 5G Core** offers a standard 3GPP interface API between 5G Core and 5G RAN (gNB).. Basic IP networking is required to reach the various components of the Private 5G Core.. A private 5G spectrum must be allocated to operate the Private 5G Core, both for indoor and outdoor operations; the Private 5G Core operates as multi-microservices and a Docker or Kubernetes architecture is required. Private 5G Core provides USIM data that need to be programmed on a USIM card with a sim card programmer. The Private 5G Core can also operate as Canonical SNAP but each component must be configured manually.

The **5G RAN** provides n-FAPI (network functional API) and FlexRAN interfaces to the 5G RAN controller components and it provides ProSe services for D2D communication APIs to the 5G D2D component. 5G RAN will operate as an independent module reachable over an IP socket interface. The IP address mapping must be known to connect to 5G RAN Intelligent controller. The 5G RIC controllers provide RAN and Network Service descriptors, respectively, to the 5G Low Latency MEC controller, which exposes 3GPP Network Slice Template (NST) API to the **Communication Resource Manager** and other external modules and components requiring to configure a 5G network. The northbound interface of the Communication Resource Manager expects as inputs the new 5G service required from the HIL service as 3GPP Network Slice Template (NST). According to 5G performance metrics or specific contexts received on its southbound interface, it will complete or alter NST before sending it to the 5G Low Latency MEC on its southbound interface, which in turn transforms this request into specific 5G RAN and CN resource configurations. This includes the slice parameters and/or other RRM configurations (e.g., scheduler, 5G numerology, transmission strategies, multi-connectivity) It also delivers as outputs the network performance measures.

In order to offer reconfiguration of 5G radio resources, the **5G RIC controller** provides a RAN API (i.e., ElasticSearch[7] technologies) to the 5G Network Monitoring component. The 5G RAN Intelligent controller sends RAN configurations to the 5G RAN component using FlexRIC API which maps a Physical Network Function (PNF) to a Virtual Network Function (VNF) using the n-FAPI API. The 5G RIC controller operates either as Docker or Kubernetes microservices. Each microservice must be assigned an IP address to be reached via IP sockets by the other microservices (components).

The **5G Network Monitoring** component will implement primarily a Kibana dashboard microservice, but a Grafana dashboard might also be used. The 5G Network Monitoring component requires an ELK stack APIs for resource monitoring from any component that it needs to monitor. The 5G Network Monitoring component is primarily based on an ElasticSearch stack and any component required to interface with the 5G Network Monitoring component should implement it.

The RESTful northbound interface of the **TSN controller** receives communication service requests and informs the requester about successful deployment of a requested communication service. It might propose alternative QoS requirements, if the requested ones are not feasible on the active network topology. A communication service request shall at least provide the following information:

1. Source MAC address
2. Destination MAC address
3. Traffic pattern, i.e., Bytes per cycle time
4. Optional Delay constraints
5. Optional send window constraints
6. Optional jitter constraints

In order to formulate a communication service request, the requesting entity, e.g., the HIL service or edge orchestrator, requires a clear definition of QoS requirements of end stations, respectively the apps being deployed

[7] https://www.elastic.co/

on end stations. The network controller indirectly interfaces with the user through the HIL service. Additionally, it offers a user interface to a network engineer, which e.g., allows to check the network state and topology. Particularly, the RESTful northbound interface provides a feature which allows a requester to exclude an endpoint from any communication, which is required for IntelIloT's moving target defences.

The southbound interface of the TSN controller communicates with TSN bridges and end systems, which are preferably configurable via netconf[8] following IEEE802.1Qcw. Additionally, legacy devices, which do not (yet) support netconf, can be integrated with device-specific southbound interface adapters, e.g., for proprietary CLI configuration.

Communication between Edge Orchestrator, Edge Manager, Edge Device Runtime, Computational Resource Manager, as well as the TSN Controller, 5G Low Latency MEC, and the Communication Resource Manager requires basic IP connectivity and uses HTTP REST APIs.

The above components, their key interfaces, brief description, and link to the implementation WP/Tasks of the project are presented in Table 6.

Table 6. Components of the infrastructure management

Component	Interface	Description	Developed in
Edge Orchestrator	EDGEO.i/f.1	Enables lifecycle operations on orchestration jobs (create, modify, delete, …). Enables onboarding of edge apps with IntelIloT specific optimization criteria. Enables service requests for the Hypermedia MAS.	T4.1
Edge Manager	EDGEM.i/f.1	Enables lifecycle operations (start, stop, install, remove, …) for edge apps on all managed edge devices. Supports mass operations, app provisioning, device onboarding and management of edge devices. (e.g., reboot, firmware update, etc.).	T4.1
Edge Device Runtime	EDR.i/f.1	Enables lifecycle operations (start, stop, install, remove, …) for edge apps on the edge device. Further it enables service functions for managing the edge device (e.g., reboot, firmware update, etc.). Difference to EM API is, that it is the corresponding API, which is offered on the concrete device. The EM API provides corresponding service functions, which includes a selection, on which devices, the function is applied.	T4.1
Computational Resource Manager	CompRM.i/f.1	Determines the optimal allocation of an Edge App In the current state of the infrastructure.	T4.1
TSN Controller	TSN.i/f.2	Computes network parameters and configures network nodes in order to guarantee QoS for requested communication services. Isolates malicious nodes.	T4.3

[8] https://datatracker.ietf.org/doc/html/rfc6241

Communication Resource Manager	NST.i/f.1	(Re)-configures the communication resources according to the traffic demands; collects network performance measures	T4.3
5G Low Latency MEC	5GMEC.i/f.1	Translates Communication Manager instructions to MEC, RAN and NVF instructions.	T4.2
Private 5G Core	OpenFlow.i/f.1	Provides interface to manage network flows.	T4.2
	3GPP.i/f.1	Provides private 5G network management.	T4.2
5G RAN	n-F.i/f.1	Provide 5G URLL/eMBB radio resources	T4.2
	FlexRIC.i/f.1	Provides interfaces to control RAN functions and send performance metrics.	T4.2
5G RAN Intelligent Controller	RAN.i/f.2	Elastic interfaces to monitor RAN and 5G RAN metrics from a remote dashboard.	T4.2
	RAN.i/f.1	Configure a 5G radio resources (slices)	T4.2
5G Network Monitoring	5GMEC.i/f.2	Provide 5G resource monitoring to external edge services.	T4.2
5G RAN Runtime	VIM.i/f.1	Corresponds to virtual resources from a virtual infrastructure manager (e.g., Kubernetes).	T4.2

2.4 Physical View

The Physical View *(how the system is deployed)* per IntellIoT component group is provided in the subsections that follow.

2.4.1 COLLABORATIVE IOT ENABLERS

Most components of theIntellIoT Collaborative IoT pillar are deployed at the IntellIoT server (see Figure 39). The **Hypermedia MAS Infrastructure** and **Global AI Component and Model Aggregator** are central to the collaboration of the multiple AI components that are active within an IntellIoT deployment (both in the multi-agent system as well as those AI components that are distributed to Edge Devices (e.g., the AI on the robot and on the tractor). These components, as well as the **IAKM Server**, require stable network connections to the different Edge Devices within the use cases while the **Interoperability Box** requires a connection to all of the Highly Constrained Devices it bridges to. Finally, the **Web-based IDE for Hypermedia MAS** as well as the **Goal Specification Frontend** are Web applications that need to be available to the Domain Engineers and Domain Experts, respectively, and require a network connection to the **Hypermedia MAS Infrastructure**.

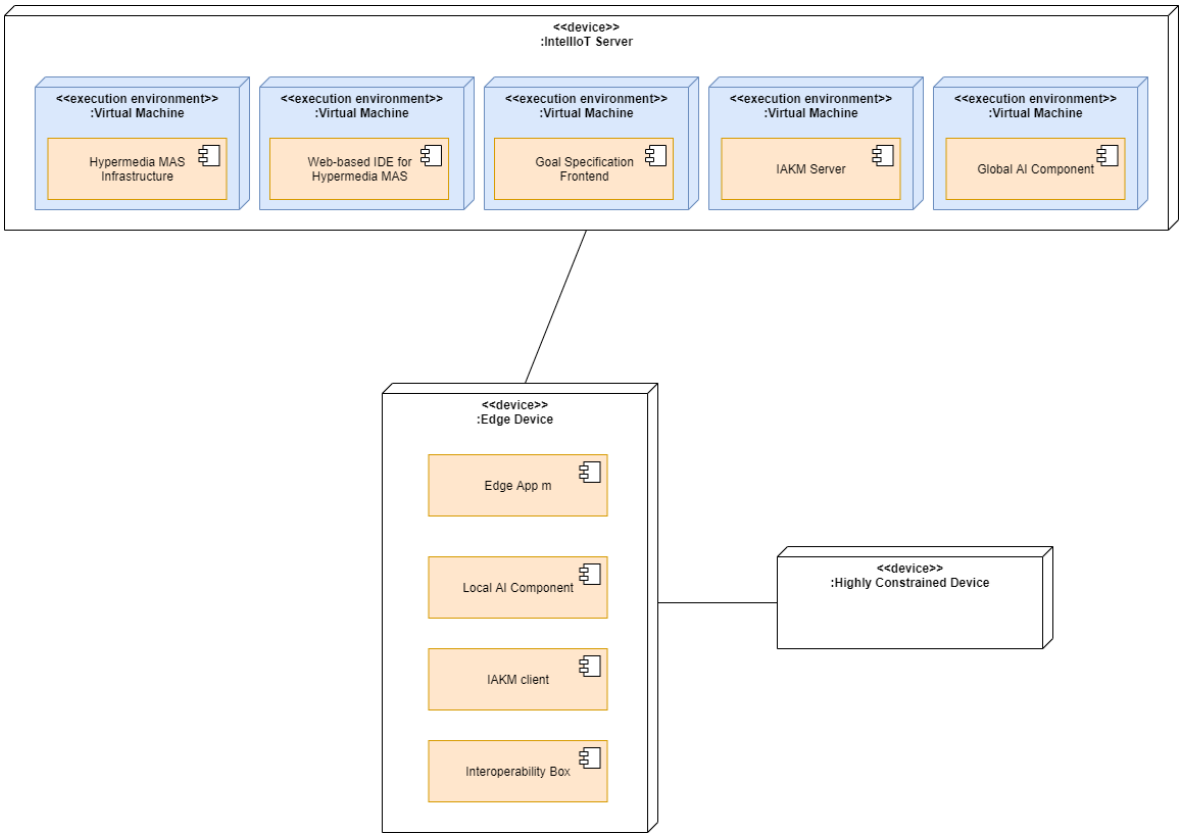

Figure 39. Collaborative IoT enablers' physical view

Table 7. Technical pre-requisites of the Collaborative IoT pillar.

Component	Technical Pre-requisites and Requirements	Developed in
Hypermedia MAS Infrastructure	Linux OS with Docker, globally reachable via HTTPS [Current Environment: i7/16GB/RTX 2060/512GB]	T3.1
Web-based IDE for Hypermedia MAS	Linux OS with Docker, globally reachable via HTTPS	T3.1
Goal Specification Frontend	Linux OS with Docker, globally reachable via HTTPS	T3.1
Infrastructure-assisted Knowledge Management	Linux OS with Docker, Python, globally reachable via HTTPS, MongoDB, MQTTS	T3.2/T4.1
IAKM Server	Linux OS with Docker, Python, globally reachable via HTTPS, MongoDB, MQTTS	T3.2/T4.1
IAKM Client	Linux OS with Docker, Python, C++ socket programming	T3.2/T4.1

Global AI Component	Python, Tensorflow 2.0, PyTorch, globally reachable via HTTPS	T3.2/T3.3
Local AI Component	Python, Java, Kotlin, Tensorflow 2.0, PyTorch, globally reachable via HTTPS	T3.2/T4.3
Interoperability box	Linux OS, globally reachable via HTTPS. Can be deployed as Docker container.	T3.1

Open Call 1 contributions

The physical view depicted in Figure 40. offers an overview of the expected deployment of the MYWAI Collaborative IoT Enablers.

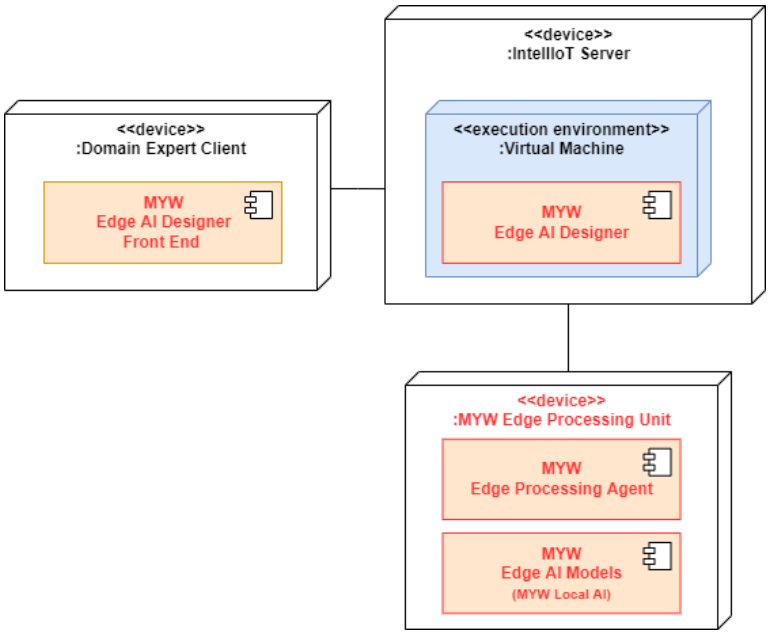

Figure 40. Physical view of the MYWAI Collaborative IoT Enablers.

The MYW Edge AI Designer appears to be suitable for deployment in the IntellIoT Server, though it can run on any server device with Windows OS.

The front end made available to the domain expert for accessing the functionalities offered by the **MYW Edge AI Designer** is a Web application suitable to be exploited from any device equipped with a browser, typically a Windows PC or a notebook.

The **MYW Edge Processing Unit** is and edge HW device designed specifically by MYWAI to run AI at the edge/very edge. With its high performance, low latency and bandwidth constraints, small footprint, and low power consumption, it enables the deployment and execution at the edge of high-accuracy AI models trained in the cloud. It can integrate CPU, GPU, FPGA, and other ASIC solutions for AI usage at the edge. Storage of data used by the local AI is managed at edge device level as well. MYW local AI models are deployed as Docker containers and can be executed virtually in any industrial edge PC.

2.4.2 HUMAN-IN-THE-LOOP ENABLERS

The HIL Service is an edge application. The **HIL Service** is triggered through an escalation by the **AI component** of the robot or tractor that is brokered by an agent in the **Hypermedia MAS**. That agent is programmed to handle the escalation by the user of the Web-based IDE for Hypermedia MAS, which requires access to the TD of the HIL service. HIL service receives the help request from the **System**. It maintains knowledge about availability, reachability and relevant properties of available **HIL Applications**. It is used to set up the connection to the supporting operator, but it is not in the loop while the operator performs his task, thus it does not have special real-time requirements. Thus, it might be deployed on any edge device.

The HIL Application is deployed on a local server, or VM, depending on the ongoing progress with the Edge Orchestrator. The Client application is already available and can be downloaded via the Microsoft Store, or the Oculus Quest Store.

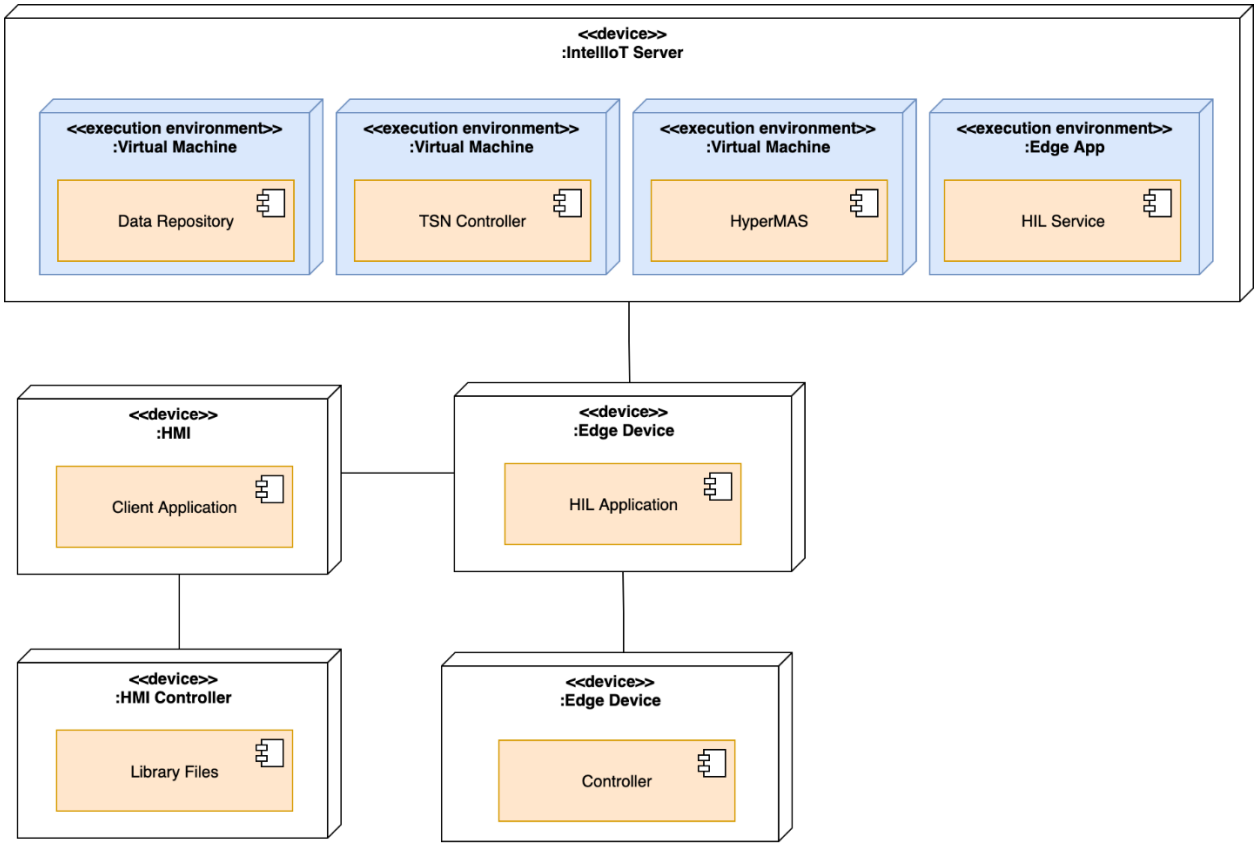

Figure 41. Physical view of HIL enablers.

Table 8. Technical Pre-Requisites of the HIL enablers software components.

Component	Technical Pre-requisites and Requirements	Developed in
Data Repository	8 Cores, > 2.4GHz CPU, 16GB RAM, > 1TB disc space for data; fault tolerance and recovery so that at least one node is always available; data models according to FHIR.	T3.2, T3.3
HIL Service	Linux OS with Docker, Java, REST, HTTPS	T4.1
Event Captor	Ubuntu 18, RabbitMQ Message Broker. Event Captor is deployed in Dockerised format with the following minimum requirements for	T4.4

	evaluation: 4 CPU Cores @ 2.4 GHz, 4 GB RAM, 8 GB disk space with up to 30 GB additional for logs.	
TSN Controller	Linux OS. Can be deployed as standalone application or containerized. Requires direct layer-2-connectivity to the network to be controlled.	T4.2, T4.3
Hypermedia MAS Infrastructure	Linux OS with Docker, globally reachable via HTTPS [Current Environment: i7/16GB/RTX 2060/512GB]	T3.1
HIL Application	Windows 10 OS. Windows min: 10.0.17763 Build 17763. NVidia Grid. Hardware Encoding. PORT 9999 TCP open. Ram: min. 16 GByte CPU: Intel I7 (7. Generation or equivalent). Graphics Card: Nvidia Geforce 10X0 with min. 5 GByte.	T 3.3
Client Application	OS Windows. Latency: (Round trip) 50ms maximum.	N/A

Open Call 1 contributions

The physical view depicted in Figure 42. offers an overview of the expected deployment of the MYWAI HIL Enablers.

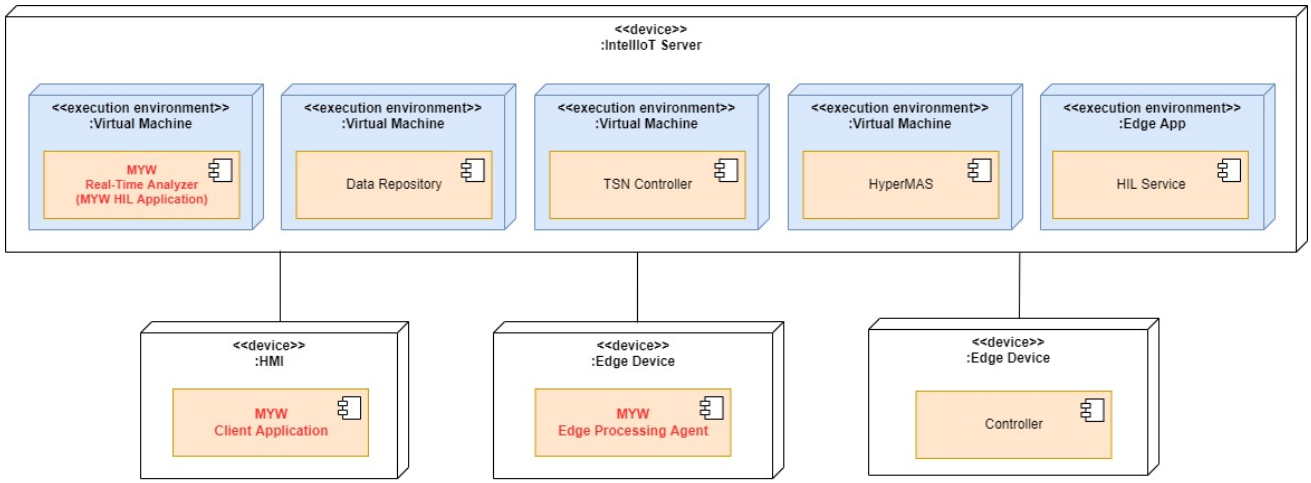

Figure 42. Physical view of the MYWAI HIL Enablers.

The MYW Edge AI Designer appears to be suitable for deployment in the IntellIoT Server, though it can run on any server device with Windows OS.

The HIL HMI for accessing to and interacting with the functionalities offered by the **MYW Real-Time Analyzer** is a Web application suitable to be exploited from any device equipped with a browser, typically a tablet or a smartphone, but also and industrial PC equipped with e.g., a touch display.

As already mentioned previously, the **MYW Edge Processing Unit** is and edge HW device designed specifically by MYWAI to run AI at the edge/very edge. With its high performance, low latency and bandwidth constraints, small

footprint, and low power consumption, it enables the deployment and execution at the edge of high-accuracy AI models trained in the cloud. It can integrate CPU, GPU, FPGA, and other ASIC solutions for AI usage at the edge. Storage of data used by the local AI is managed at edge device level as well. MYW local AI models are deployed as Docker containers and can be executed virtually in any industrial edge PC.

2.4.3 TRUST ENABLERS

On the physical layer, the trustworthiness software components can be divided to those deployed on edge devices, and those deployed on a server, or the cloud (Figure 43). Depending on the component, the hardware on which it is depicted to be deployed on is not binding. Several different options may be available (for example see technical pre-requisites and requirements for the Security Assurance Platform in Table below), but only one of them is shown in the diagrams, according to the selection made during the design of the IntellIoT Use-Cases to be demonstrated.

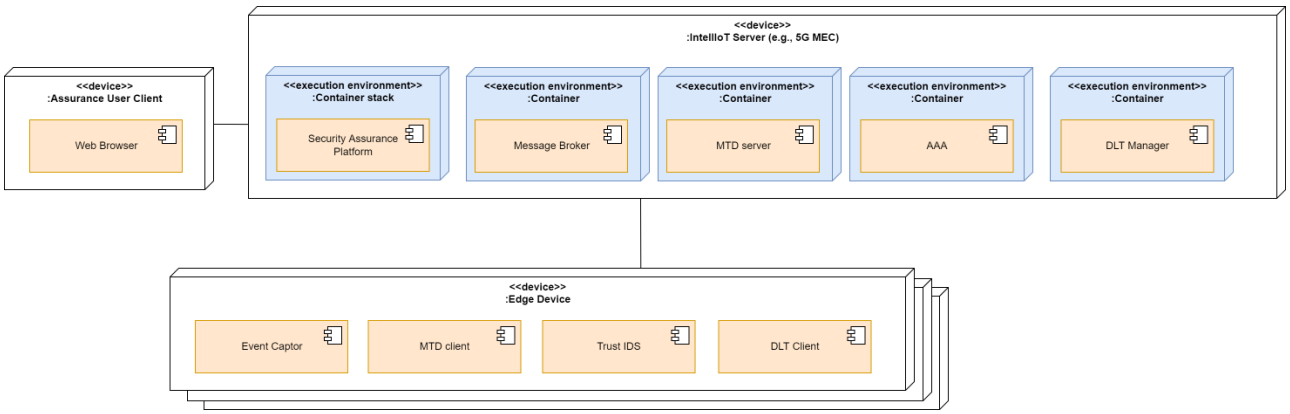

Figure 43: Physical view of the Trustworthiness components.

The Event Captors, the IDS, the MTD Client and the DLT Client are all deployed on each edge device of the IntellIoT framework, with a few exceptions (for example there are certain restrictions on Smartphones' OS level that do not allow these components to be deployed or function properly). These can be deployed natively or in containers (dockerised form) and do not need to be set up inside a Virtual Machine (VM). On the other hand, the Security Assurance Platform, Message Broker, MTD Server, AAA module, and the DLT Manager are all deployed in their respective containers (or, alternatively VMs, depending on the intricacies of each deployment), which in turn exist on IntellIoT's infrastructure. Finally, the Security Assurance Platform's frontend is also depicted, which is accessible through any modern Web Browser.

Table 9. Technical pre-requisites of the security & trust components

Component	Technical Pre-requisites and Requirements	Developed in
Security Assurance Platform	Can be used in cloud, on-site or hybrid modes. Can be deployed locally, on VMs or in Dockerised form (currently requiring Docker v20.10.7 & Docker-compose v1.27.4). Components require in total (approximately) the following: Minimum: CPU w. 4 cores, >=16GB RAM, >=100GB SSD storage, 10/100MB network connectivity Recommended: CPU w. 8 cores, >=32GB RAM, >=500GB SSD storage, Gigabit network connectivity. Connectivity requirements (exposed ports): 443, 8245	T4.4
Event Captor	Ubuntu 18, RabbitMQ Message Broker. Event Captor is deployed in Dockerised format with the following minimum requirements for evaluation: 4 CPU Cores @ 2.4 GHz, 4 GB RAM, 8 GB disk space with up to 30 GB additional for logs.	T4.4

Message Broker	Can be deployed in Dockerised format or locally, with the following minimum requirements: 128 MB RAM, 2GB disk space, x86 (64-bit) CPU architecture.	T4.4
AAA	Can be deployed in Dockerised format. It consists of two containers, a Keycloak and a MySQL. Minimum requirements: 1GB RAM, 1GB of disk space, x86 (64-bit) CPU architecture.	T4.4
Intrusion Detection System Instance	Linux OS. Needs administrator access to Linux Network Stack. Can be deployed as privileged Docker container, or locally. Minimum requirements: 1-2 CPU cores @ 2.4 GHz x86 (64-bit), 2 GB RAM, 20 MB Storage for the installation.	T4.4
Moving Target Defences (Server)	Linux OS. Can be used in cloud, or on-site. Can be deployed as Docker container, inside a VM, or locally. Minimum requirements: 1-2 CPU cores @ 2.4 GHz x86 (64-bit), 2 GB RAM, 20 MB Storage for the installation.	T4.4
Moving Target Defences (Client)	Linux OS. Needs administrator access to Linux Network Stack. Can be deployed as privileged Docker container, or locally. Minimum requirements: 1-2 CPU cores @ 2.4 GHz x86 (64-bit), 2 GB RAM, 20 MB Storage for the installation.	T4.4
TSN Controller	Linux OS. Can be deployed as standalone application or containerized. Requires direct layer-2-connectivity to the network to be controlled.	T4.2, T4.3
DLT Manager	Linux OS, IPFS storage, MQTT broker, can be Dockerised or implemented locally. Minimum requirement: Memory 4GB, processor Intel® Core™ i7-3517UE CPU @ 1.70GHz × 4, x86(64-but) CPU architecture, internet connection.	T3.4
DLT Client	Linux OS or Windows can be Dockerised format or implemented locally. Minimum requirement: memory 1GB, 1 CPU cores @2.4 GHz, 1GB RAM, 8GB disk space, work both locally and via internet.	T3.4

2.4.4 INFRASTRUCTURE MANAGEMENT

The physical view of the infrastructure components is shown in Figure 44. The components for the management of the edge orchestration are physically located on a dedicated machine (left side of the figure). The counter part of the Edge Manager, the Edge Runtime component, is located on each single edge device. An edge device is a computing resource on the edge infrastructure that can host multiple Edge Apps as shown on the right side of the figure. A set of Edge Apps that are currently planned is shown as being hosted by "Edge Device 1". In an actual deployment, those would be dispersed over multiple Edge Devices. Not only use case functionalities, but also infrastructure management components (e.g., TSN Controller, Communication Resource Manager, etc.) will be deployed as Edge Apps. The 5G system will be again deployed on a separate machine, as shown at the bottom of the figure.

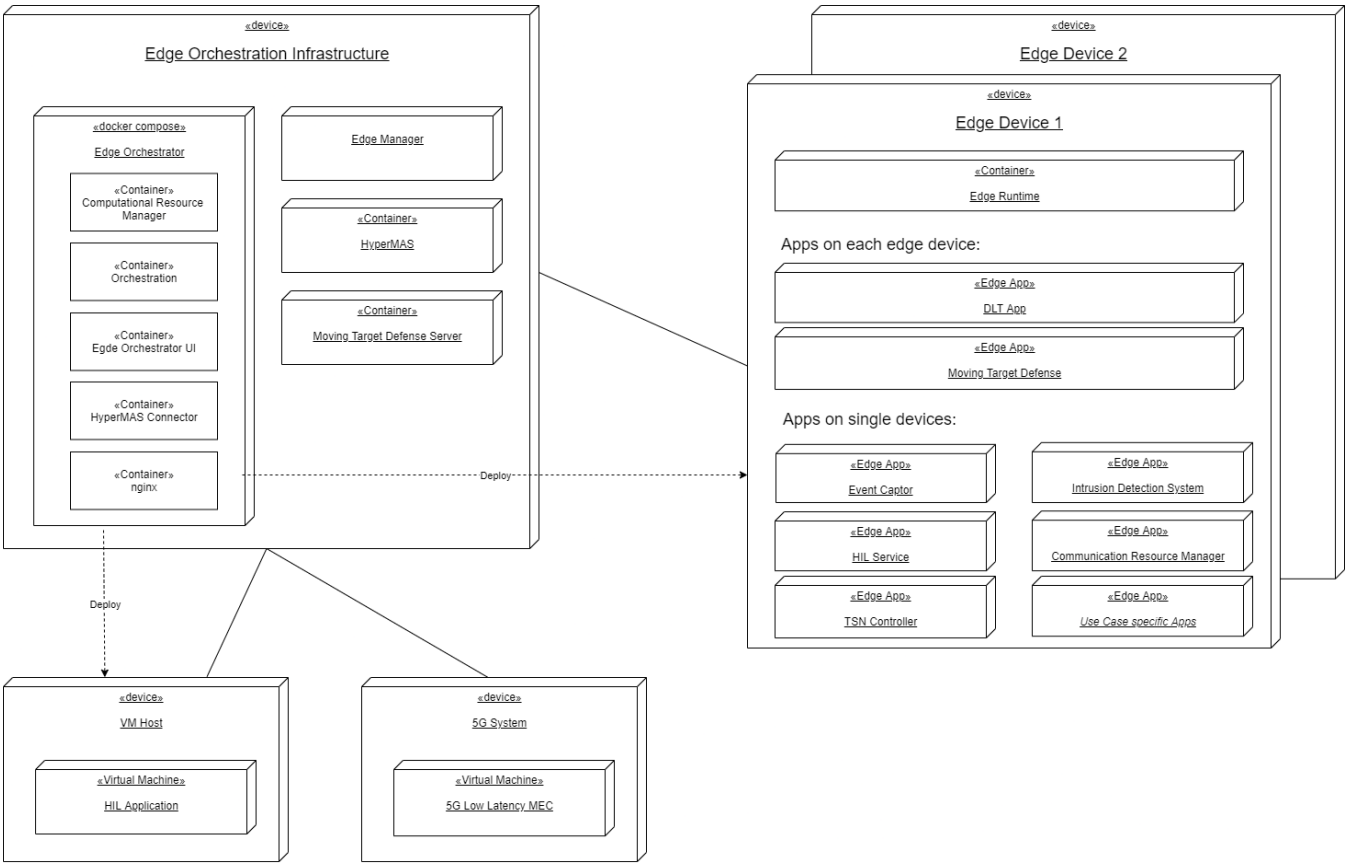

Figure 44: Physical view of the infrastructure management components.

The technical pre-requisites and requirements for each software component are listed in Table .

Table 10. Technical pre-requisites of the infrastructure management components

Component	Technical Pre-requisites and Requirements	Developed in
Edge Orchestrator	Has to be deployed on an industrial PC (x86, 4 CPU Cores @ 2.4 GHz, 8 GB RAM, 256 GB HDD) in a Linux environment, where docker and docker-compose is available.	T4.1
Edge Manager	Has to be deployed on an industrial PC (x86, 4 CPU Cores @ 2.4 GHz, 16 GB RAM, 256 GB HDD) in an environment where VMWare Workstation is available	
Edge Device Runtime	Component has to be hardware platform, which supported by Siemens Industrial Edge (e.g., SIMATIC Nano).	
Computational Resource Manager	Component will be deployed in dockerized format with the following minimum requirements for evaluation: 4 CPU Cores @ 2.4 GHz, 8 GB RAM.	T4.1
TSN Controller	Linux OS. Can be deployed as standalone application or containerized. Requires direct layer-2-connectivity to the network to be controlled.	T4.2, T4.3
Communication Resource Manager	Off-the-shelf PC (x86, 4 CPU Cores, 16 GB RAM, 256 GB HDD) on Linux environment. It needs to support docker, docker-compose and Snaps. Can operate on the same PC as 5G LL MEC as separate Docker Container.	T4.3

5G LL MEC	Off-the-shelf PC (x86, 4 CPU Cores, 16 GB RAM, 256 GB HDD) on Linux environment. It needs to support docker, docker-compose and Snaps.	T4.3
Private 5G Core	Off-the-shelf PC (x86, 4 CPU Cores, 16 GB RAM, 256 GB HDD) on Linux environment. It needs to support docker, docker-compose and Snaps. Can operate on the 5G RAN PC or on the 5G LL MEC PC as separate docker container.	T4.3
5G RAN	Off-the-shelf PC (x86, 4-6 CPU Cores, 32 GB RAM, 256 GB HDD) on Linux environment. It needs to have USB3 connector to connect to the Radio front-end. USRP or AWS radio front-ends connected on USB3.	T4.3
5G RAN Controller	Off-the-shelf PC (x86, 4 CPU Cores, 16 GB RAM, 256 GB HDD) on Linux environment. It needs to support docker, docker-compose and Snaps. Can operate on the same PC as 5G LL MEC as separate docker container.	T4.3
5G Network Monitoring	Off-the-shelf PC (x86, 4 CPU Cores, 16 GB RAM, 256 GB HDD) on Linux environment. It needs to support docker, docker-compose and Snaps. Can operate on the same PC as 5G LL MEC as separate Docker Container.	T4.3

2.5 Use-Cases (+1) View

The Use Cases View *(how the system is applied in different use cases)* per IntellIoT component group is provided in the subsections that follow.

2.5.1 UC1 – AGRICULTURE

The agriculture use case is deploying the majority of the components from the generic IntellIoT architecture, but explicitly targeting them to the agricultural domain (see Figure 45).

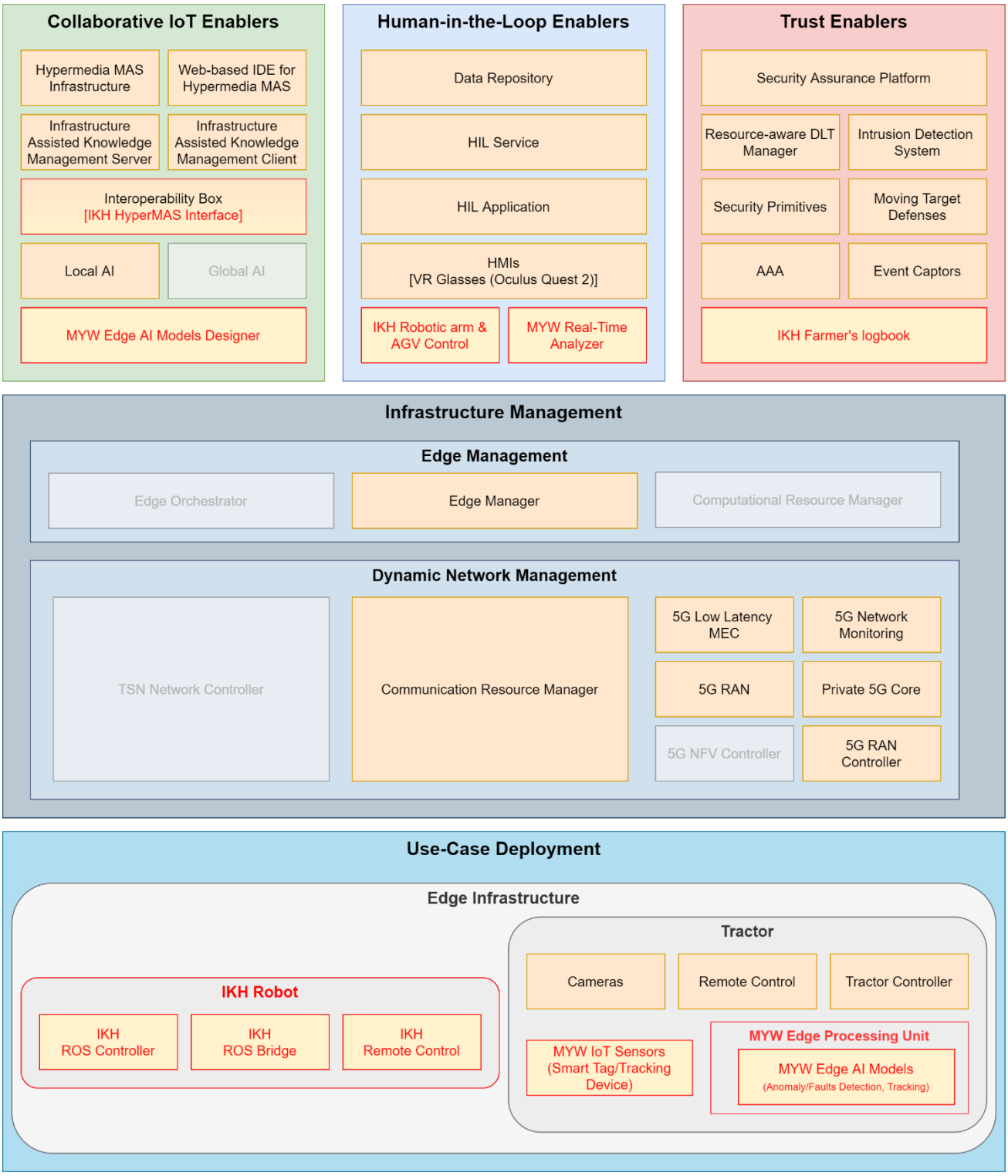

Figure 45. The IntellIoT framework applied on the Agriculture Use Case

In more detail:

Collaborative IoT Enablers

Within the collaborative IoT enablers, the Hypermedia MAS will enable the human operator to identify the different vehicles (e.g., tractors or harvesters) available for the farmers and plan them for a specific mission (e.g., harvesting

or ploughing). Additionally, it will offer the possibility to select the specific field, where the mission needs to be executed and how the mission will be performed. During the execution of the mission, it can happen that a vehicle is stuck in a specific situation (e.g., unknown obstacle) and it doesn't have the correct knowledge to overcome the situation. The Infrastructure Assisted Knowledge Management (IAKM) will support the vehicle to identify other entities participating in the knowledge scheme (e.g., other tractors) that could have the correct knowledge and provide it to the tractor to overcome the situation. Additionally, the local AI will also support the overcoming of unknown situations. Based on the data received from the sensors from the tractor, the local AI will calculate suggestions for bypassing the situation (e.g., an obstacle). If the suggestion is a viable one, the tractor will use the suggestion. If this doesn't provide a viable solution, the IAKM will try to locate an updated model and if that is not available, the human operator will be contacted (see Human-in-the-Loop Enablers).

Human-in-the-Loop Enablers

Within the Human-in-the-Loop enablers the human operator will have the possibility to directly interact with the tractor via a 5G connection by receiving data streams from and commanding driving commands to the tractor. The HIL Application in combination with the HIL Service and the Agent enable this. The HMI for the human operator consists of VR glasses (Oculus Quest 2) displaying a Virtual Reality (VR) view and a controller (e.g., game controller) for sending actuation commands directly to the **Tractor Controller** (see below) and drive the tractor around the unknown obstacle that is blocking its mission. The data from the actuation is then stored in the data repository and the local AI from the collaborative IoT enablers are used to learn new knowledge to overcome similar situation in the future.

The most important component for the HIL application is the **Tractor Controller**. It is available for the tractor and will have multiple interfaces enabling the interaction, including but not limited to the sensor boxes from the tractor. It will provide e.g., Ethernet, CAN, USB and RS232 interfaces for connecting tractor subsystems, like the steering actuators and the electric motors of the vehicle. The tractor controller will mainly host the different agriculture edge apps (e.g., MTD Client, DLT Client, etc.) developed by the other partners inside the use case and it is still under definition what kind of software interfaces will be applied, based on the edge operating system running on the tractor controller.

Trust Enablers

Within the trust enablers, the IDS monitors the tractor and the drone, and warns the Security Assurance Platform in case of unexpected behaviour. The Security Assurance platform activates the MTD when necessary, and the MTD mitigates the attacks. This ensures that no external malicious entity can take over the entities in the field and e.g., starts sending falsified information or even take over the control of the vehicle.

Dynamic Infrastructure Management

The Edge Manager will take care of the actual deployment of the application on the vehicle (i.e., edge device on the vehicle) itself. The Computational Resource Manager will not be deployed inside the agriculture Use Case, as each entity in the field (i.e., a vehicle) will have in principle all applications hosted on its controller and there will be optimization targeted for deploying apps on different entities.

No TSN controller will be deployed in this use case, as no deterministic communication is currently required for the functionality of the vehicles. The communication resource manager will allocate radio resources to UEs depending on their needs to ensure bandwidth and latency guarantees. This is realized through network slicing, where a slice can be created for specific sets of UEs to isolate a set of resources for only them to use. This allows guarantees to certain traffic flows, like VR, for specific UEs to not be delayed and arrive timely, even if other traffic occurs meanwhile.

Agriculture UC Deployment

Within the agriculture UC, the tractor (or more specifically the tractor controller) can be viewed as the edge device. It will host multiple applications, varying from the control of the tractor (e.g., driving, steering), but it will also have access to the data stream coming from the different sensors mounted on the vehicle. Additionally, within the use case, the addition of a drone is targeted (potentially coming from the open calls), which can also be viewed as a separate edge device and is additionally a UC specific deployment.

Open Call 1 contributions

Open Call 1 Winner **IKnowHow (IKH)** contributed to the agriculture use case with different components that are positioned at different locations in the infrastructure (as depicted in Figure 45). A specific interface to the Hypermedia MAS has been created to enable the interaction between their system and the Hypermedia MAS. Furthermore, a control with the VR solution has been established to perform Human-Machine-Interaction between the robot (the moving platform) and the human as also between the robotic arm mounted on the moving platform and the human, enabling IntellIoT to also offer a remote operation for robotic arms through VR. Additionally, based on the DLT solution available in the architecture, a Farmer's Logbook has been integrated, which records the actions of the robotic platform.

Finally, the robotic platform from IKH has been integrated as an additional edge device in the infrastructure on the use-case deployment level. It contains a ROS Controller, a ROS Bridge and a special remote control to perform remote operation of the robotic platform if needed. All these new additions to the +1 view of the agriculture use case have been included in the overall IntellIoT architecture.

Furthermore, Open Call 1 Winner **MYWAI** provide their own set of enablers. The mechanical/electrical components of the tractor and/or other agricultural machinery may break, show signs of wear or malfunctions. The MYWAI Enablers will allow automatically detecting anomalies, malfunctioning and failures through AI-based algorithms running on an edge/very edge device and applied to data time series gathered by heterogeneous sensors placed on the monitored equipment. Anomaly/failure models will be designed by domain experts according to available real time and/or historical data collected by the sensors. Detected anomalies/failures will be promptly notified to human operators to trigger their intervention, either from remote or on the field.

2.5.2 UC2 – HEALTHCARE

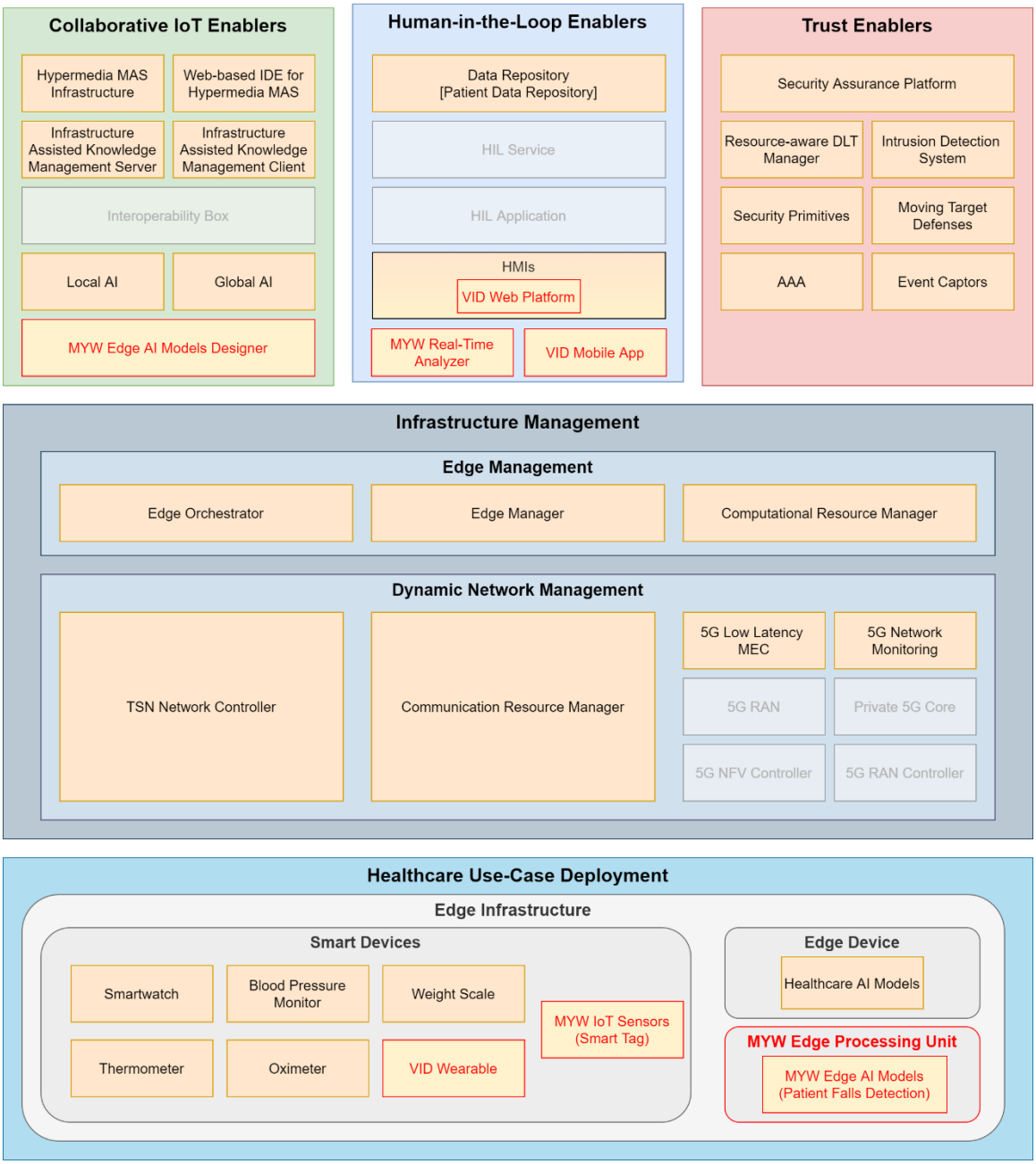

Figure 46. The IntelllIoT architecture applied on the Healthcare Use-Case.

The healthcare use case is deploying the majority of the components from the generic IntelllIoT architecture, but explicitly targeting them to the healthcare domain (see Figure 46).

In more detail:

Collaborative IoT Enablers

Within the collaborative IoT enablers, the Hypermedia MAS will function as a thin layer enabling the communication of physician-approved interventions to the patients. Due to the sensitive content of the interventions produced by the AI, these will have to be approved by a physician before they are sent to a patient. Also, in some cases the AI will produce inconclusive interventions at which point Hypermedia MAS will handle escalation to the HIL service, which in turn will involve human assistance from a physician. Meanwhile, the Global AI, Local AI, and Healthcare AI Models together form the federated learning system. The system ensures that data remains private, as AI model training is performed by the Local AI on local datasets in Edge devices and model aggregation and evaluation is

performed by the Global AI in the 5G MEC. A number of base or pretrained Healthcare AI Models will produce the mentioned interventions. Healthcare AI models can be shared between global AI through the IAKM server.

Human-in-the-Loop Enablers

Within the Human-in-the-Loop enablers, the physician will have the possibility to send interventions to patients. A password-protected web interface accessible through a laptop or mobile device will show possible interventions from which the physician can select which one to send to a patient. Some of the interventions will be produced by the base or pretrained Healthcare AI Models, which are hosted in the Global AI and also initially in the Local AI. Once the Local AI has trained the Healthcare AI Models on local datasets specific to a patient, some interventions can be personalized for that patient. The local datasets are produced by the sensors of the Healthcare UC that take different measurements of a patient, such as body temperature, body weight, blood pressure, and heart rate. In addition, the human-in-the-loop and the IntelIIoT framework make it possible to use the selected intervention to train the AI models further with this new labelled dataset, which is otherwise extremely scarce and difficult to obtain. The Patient Data Repository will be implemented in the FHIR[9] standard, will collect patient data that is needed by the physicians, and will be connected to the web interface.

Trust Enablers

The Trust enablers will ensure the privacy of the patient's data and integrity of the devices involved, both on the Patient Data Repository and their IoT devices. To this end, Event Captors will be deployed on the Data Repository located in the hospital, where they will continuously monitor file access events and report any intrusions to the Security Assurance Platform. These intrusions can happen as a result of a successful hacking attempt or by a simple system misconfiguration. Malicious attacks however can also happen at the patient's premises, targeting their smart devices. Deploying Event Captors on proprietary smart devices is not as straightforward; however, event captors can be deployed at the patient's gateway, to monitor their traffic and detect botnet attacks.

Dynamic Infrastructure Management

The Dynamic Infrastructure Management will be reduced in this UC, due to the use of commercial 5G networks. A 5G MEC, cloning a real 5G MEC entity in a 5G private network, will be hosted in a private cloud and connected to edge devices and hospital backend via commercial 5G network.

Healthcare UC Deployment

Within the healthcare UC, five smart devices will be used as sensors to monitor patients and collect measurements: a smartwatch, a weight scale, a thermometer, an oximeter, and a blood pressure monitor. The smart devices will sync the collected data to the Edge device, which will be a smartphone where the Local AI can train Healthcare AI models on local datasets.

Open Call 1 contributions

Open Call 1 Winner **Vidavo** implements the Healthcare use case that aims to support CVD patients that following specific care plans related to their physical activities and conventional therapies. The care plans are created for the Web interface, where physicians state the type of physical activities, the duration and frequency that should be followed. Besides that, physicians include to care plans requirements related to medical exams that patients should take. The workflow of medical exams and care plans are supported from a mobile application that patients use. The mobile application provides the perfect user interface for patients to:

- Perform and communicate with medical devices and automatically send the exams to physicians
- Process locally the medical data and create alerts and notification based on physicians' guidelines
- Let physicians know if the care plan was executed successfully or not
- Report their symptoms with tailored-made questionnaires
- Present medical data and their progress over time
- Execute machine learning models without any internet connection in order to predict outcomes related to physical activities

[9] https://www.hl7.org/fhir/overview-arch.html

The web application is used from physicians to: Create patients' profile; Insert patients' data regarding baseline ECG and Cardiopulmonary exercise test (CPET); Create care plans and monitor their progress; Receive notification based on their guidelines; Receive notification from the AI models to be approved; Receive monthly reports highlighting patients' progress and issues, and; Monitor data regarding medical exams related to blood pressure, oximetry, body composition and temperature. An overview of these offerings is provided in Figure 47.

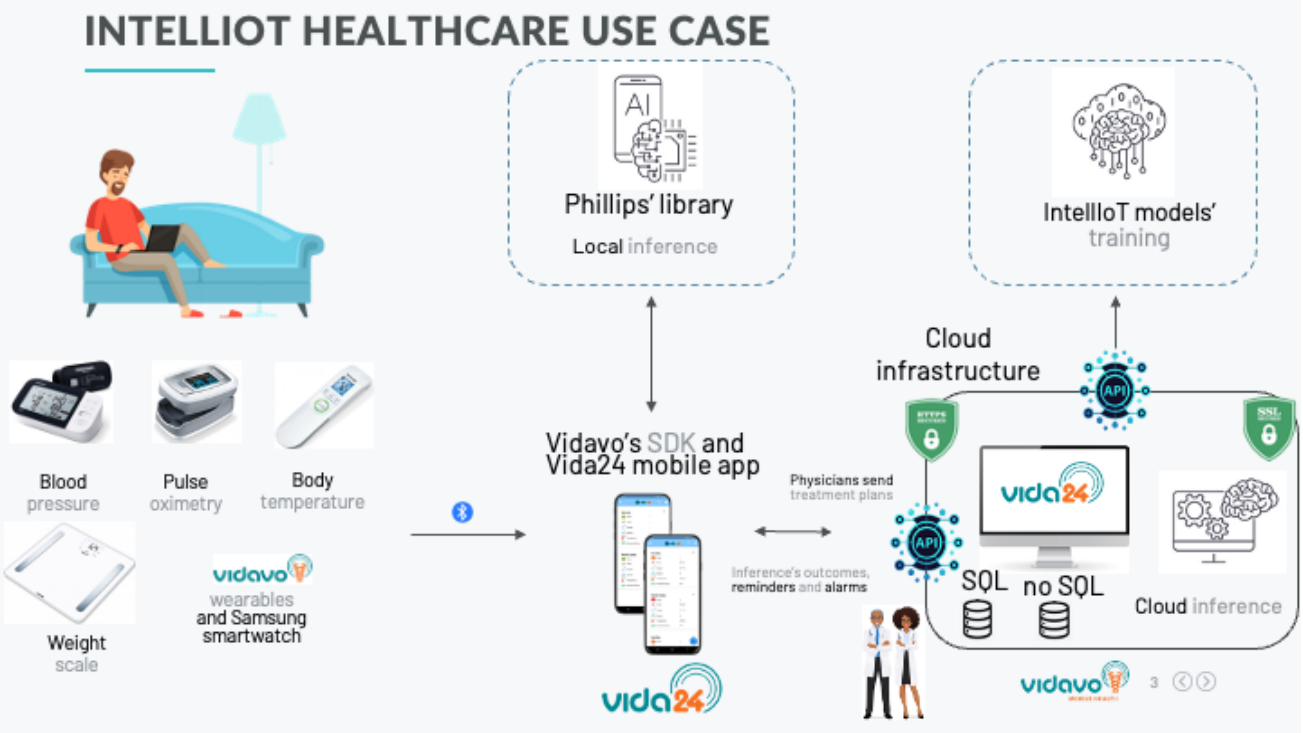

Figure 47. Vidavo enablers within IntelIIoT's 2nd Use Case.

Open Call 1 Winner **MYWAI** brings its own set of enablers into the 2nd Use Case. Patients with cardiovascular disease are subject to a high risk of falling (e.g., pre-syncope dizziness, pressure changes, arrhythmia, etc.). More generally, this is a risk that characterizes the elderly with often harmful consequences in the event of a fall. The MYWAI Enablers will allow automatically detecting postural anomalies, wrong movements and falls through AI-based algorithms running on an edge/very edge device at the patient's home and applied to data time series gathered by wearable sensors (e.g., wrist-worn and/or belt accelerometers). Behavioural anomaly/fall models will be designed by domain experts according to available real time and/or historical data collected by the sensors. Detected behavioural anomalies/dangerous situations/falls will be promptly notified to clinicians, care givers and/or patient's family to trigger their intervention, either from remote or directly at the patient's home.

2.5.3 UC3 – MANUFACTURING

The manufacturing use case is deploying the majority of the components from the generic IntelIIoT architecture, but explicitly targeting them to the manufacturing domain, as shown in Figure 48.

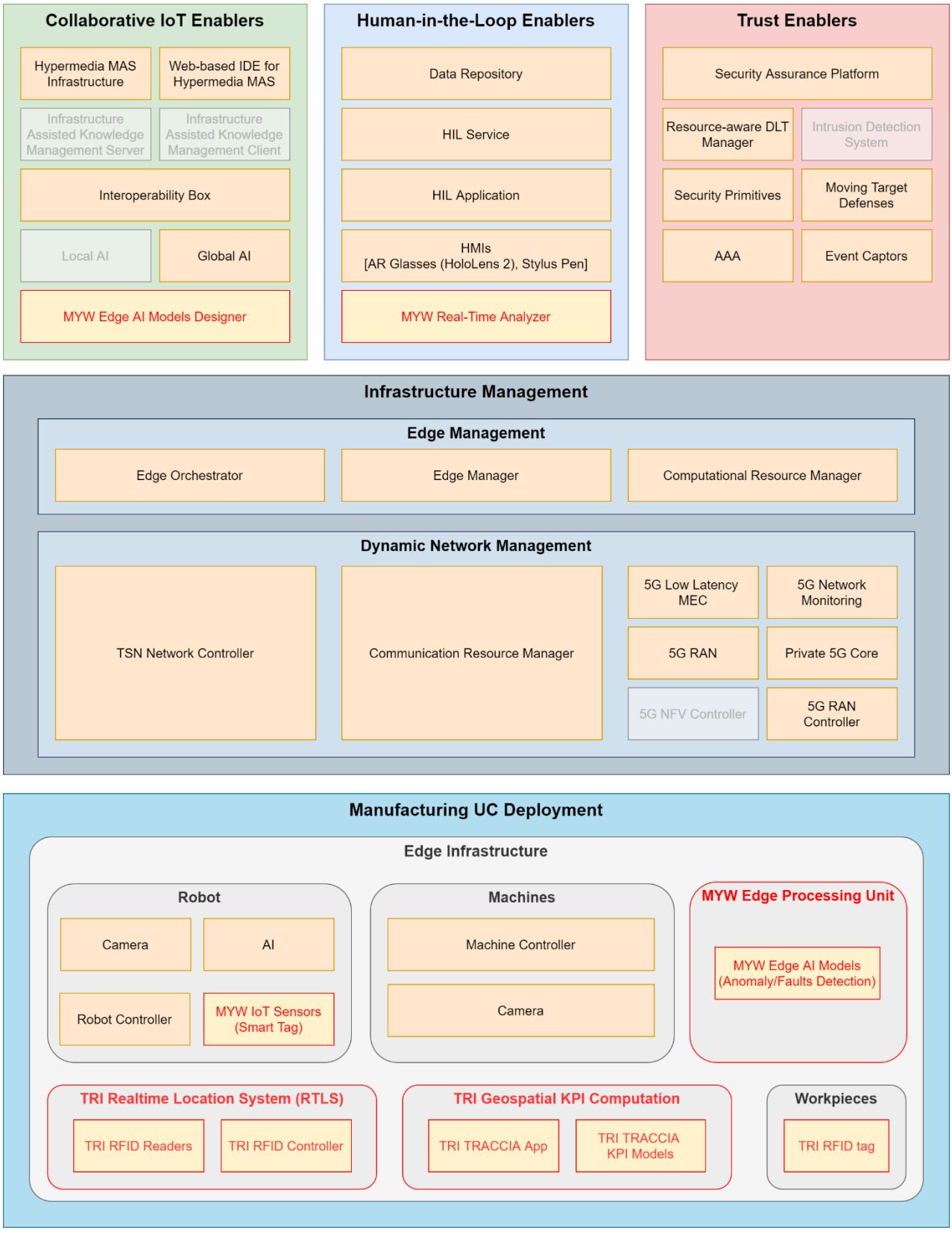

Figure 48. The IntellIoT framework applied to the Manufacturing Use Case

In more detail:

Collaborative IoT Enablers

Within the collaborative IoT enablers, the Web-based IDE for Hypermedia MAS will enable the customer to specify a goal, i.e., to place an order for a personalized product. During production of the product, it might happen that AI is unsure about a decision, e.g., how to grab a workpiece. Hypermedia MAS will escalate this issue to the HIL service in order to get human assistance.

Human-in-the-Loop Enablers

Within the Human-in-the-Loop enablers the human operator will have the possibility to directly interact with the robot. AR glasses will show him the scene the robot is working in, streamed through a camera mounted on the robot. The operator will be able to control the robot with the help of a stylus input device. The HIL application, invoked by the HIL Service, enable this. After a successful HIL operation, the current robot position, which now defines the desired grab spot, will be stored together with the camera image. Both are used to re-train AI from the collaborative IoT enablers in order to overcome similar situation in future.

Trust Enablers

Within the trust enablers, the IDS monitors the robot and machines, and warns the Security Assurance Platform in case of unexpected behavior. The Security Assurance platform activates the MTD when necessary, and the MTD mitigates the attacks. To do so, it might command TSN controller to isolate malicious nodes. This ensures that no external malicious entity can take over the entities in the field and e.g., spy out confidential information using the camera on the robot.

Dynamic Infrastructure Management

The manufacturing use case is the main application for edge management in IntellIoT. The Edge Manager in cooperation with computational resource manage will take care of the actual deployment of edge applications on edge devices. With the support of the edge manager, the Hypermedia MAS will be enabled to invoke applications which are not yet deployed in the system in advance.

The TSN controller for wired communication as well as communication resource manager for wireless communication will be invoked by the edge orchestrator in order to establish a tactile communication between a robot requiring help and an available human operator.

Manufacturing UC Deployment

Within the manufacturing UC, hardware nodes comprising computational resources are treated as edge devices. Edge apps are deployed on edge devices by the edge manager, while the computational resource manager maintains the optimal allocation. For some applications, e.g., robot and machine controllers, communication requirements are to be considered and are maintained by the TSN controller and the communication resource manager. For these kinds of applications, restrictions for the deployment may arise from communication requirements, particularly on delay.

Open Call 1 contributions

The manufacturing process will be monitored by the RTLS system, provided by Open Call 1 winner **Trilogis (TRI)**, based on RFID technology. Each workpiece that is processed by the different machines (e.g., milling machine, engraver) is tagged with a RFID tag to supervise the displacement of each workpiece and analyze the efficiency of each working phase. Toward this end, different proximity areas will be supervised by the RFID readers. Each RFID reader is managed by an RFID controller. The software platform TRACCIA (web application, background IPR of Trilogis) is configured to process the workpiece positions and compute the geospatial KPIs related to the manufacturing process. The KPIs are defined in TRACCIA using the available position data and the information on the process (e.g., the status of the machines involved in the manufacturing will be provided by Hypermedia MAS through the dedicated APIs)

Open Call 1 winner **MYWAI (MYW)** bring their own set of enablers into the use case. The mechanical/electrical components of the robot arm and/or other manufacturing equipment may break, show signs of wear or malfunctions. The MYWAI Enablers will allow automatically detecting anomalies, malfunctioning and failures

through AI-based algorithms running on an edge/very edge device and applied to data time series gathered by heterogeneous sensors placed on the monitored equipment. Anomaly/failure models will be designed by domain experts according to available real time and/or historical data collected by the sensors. Detected anomalies/failures will be promptly notified to human operators to trigger their intervention, either from remote or in the industrial plant.

3 DERIVED TECHNICAL REQUIREMENTS

In Deliverable D2.2 (Section 4) the Functional and Non-Functional Requirements of the IntellIoT framework have been defined. According to this definition, these requirements cover the set of user-visible requirements, that is they describe user visible functionalities and qualities that the IntellIoT framework must, should or may provide. However, another set of requirements, named *Technical Requirements*, has been identified. These requirements cover functionalities, organization and architecture issues as well as component features that are internal to the functioning of the framework and while they are not visible to an external user, they are fundamental for the proper functioning of the system that IntellIoT plans to develop. This section lists these specific requirements.

Similarly, to the requirements described in Deliverable D2.2, the Technical Requirements presented in Table 11, have a unique associated ID (TR.#) to make them identifiable throughout the duration of the project, a requirement description and a level characterization. The latter has the MUST/SHOULD/MAY rating, denoting requirements that crucial and obligatory to be implemented, welcome features and optional requirements respectively.

Table 11. IntellIoT's technical requirements

ID	Requirement	Level	Changes to D2.3
TR.1	When a service registers to the Hypermedia MAS Infrastructure by providing a W3C WoT TDT, this data structure shall include at least: - A JSON-LD Context and Type - A Title and Description - The structure of inputs and outputs (for mapping within concrete agent programs) - For any property, action, or event affordance, a human-readable description, a reference (templated Id) to the service representation at the Edge Orchestrator.	MUST	
TR.2	When a service registers to the Hypermedia MAS Infrastructure by providing a W3C WoT TD, this data structure shall include, in addition to the information in the TDT that is specified above: - A valid protocol binding (i.e., forms field) for HTTP, CoAP, or MQTT - A security definition (i.e., security definitions and security fields)	MUST	
TR.3	When an agent intends to use the service of an edge app that is represented through a W3C WoT TDT, it sends a create orchestration request to the Edge Orchestrator, who, in turn, instantiates the service and registers a protocol-bound W3C WoT TD at the Hypermedia MAS.	MUST	
TR.4	While a service is available, it shall acknowledge heartbeat requests by the Hypermedia MAS Infrastructure.	SHOULD	Updated level.
TR.5	When the Hypermedia MAS receives a goal, the Hypermedia MAS shall delegate this goal to the agents according to the configured agent organization and the agents shall start working on the goal.	MUST	
TR.6	When agents task services (including services that shadow devices such as vehicles, robots, and machines), they shall	MUST	

	send task information to these services using the API and formats that are specified in the services' TDs.		
TR.7	When a task has been successfully received by a service, the service shall send out a confirmation to the agent that tasked it and shall start executing the assigned task.	MUST	
TR.8	When a request for ML-related knowledge has been received by the IAKM, the IAKM shall contact potential entities if they can provide the requested information.	MUST	
TR.9	When the AI can calculate control decisions (linear and angular velocities) with high confidence, it must provide those decisions to the tractor controller to support the obstacle bypassing operation.	MUST	Updated to make it more specific.
TR.10	Each entity in the field must host a version of a distributed ledger	MUST	
TR.11	Each entity in the field must host a smart contract application via DLT APIs	MUST	
TR.12	When an agent tasks a service (including services that shadow vehicles, robots, and machines), the agent shall provide the task information to the DLT.	MUST	Removed. Redundant with TR.24
TR.13	The environmental data must consist of one or more videos streams from the tractor.	MUST	
TR.14	When requesting a transport job from a robot, an agent shall deliver the following information according to the robot TD: - ID of workpiece - Source Machine / Source Location - Target Machine / Target Location	MUST	Removed. This is covered by the adherence to W3C WoT TD in GNFR.18.
TR.15	Every machine and robot shall provide a REST-service interface to allow Hypermedia MAS to poll its current state. The information provided by this interface shall include at least the following data: - Machine / Robot ID - Task ID - Progress information	SHOULD	Updated as the logic of polling was changed from the machines actively pushing updates to the Agents pulling updates.
TR.16	When a robot grabs or places a workpiece, it shall inform the Hypermedia MAS Infrastructure and the DLT. The information shall include the following data: - Machine ID - Task ID - Event description	MUST	Removed. Architecture reworked. Switched from REST to polling. See TR.15.
TR.17	When a machine starts or finalizes a task, it shall inform the Hypermedia MAS Infrastructure and the DLT about its progress on the task. The information shall include the following data: - Robot ID - Task ID	MUST	Removed. Architecture reworked. Switched from REST to polling. See TR.15.

	- Event description		
TR.18	When a machine or robot takes a task, it shall inform the Hypermedia MAS Infrastructure and the DLT about taking the task. The information shall include the following data: - Machine / Robot ID - Task ID	MUST	Removed. Architecture reworked. Switched from REST to polling. See TR.15.
TR.19	When a DLT client setup in robot or machine controller wants to record data to the ledger, the smart contract application shall receive the transactions from machines and robots regarding tasks, which includes IDs and progress.	MUST	
TR.20	When the Smart Contract service starts up, it shall register to the Hypermedia MAS and account the output of generated tasks.	MUST	
TR.21	The TCP ports 9999 as well as ports 8080-8099 on the test site must not be blocked by a Firewall. The secure MQTT port 8883 must not be blocked by a Firewall.	MUST	
TR.22	The AI on the robot should provide additional information about available workpieces to the Hypermedia MAS. This information should include the following information: - Height, width, centre and orientation of the rectangle with the largest area which can be placed on the workpiece. - Height, width, centre and orientation of the rectangle with the largest width which can be placed on the workpiece. - Additional information for future use, e.g., material and thickness.	SHOULD	Removed. No responsibility on data collection and labelling and thus, training new models is not possible.
TR.23	All components comprising computational resources, e.g., robots, agents and customer interface devices, shall run a smart contract application and DLT client on their local computational resources, to minimize attack vectors against the DLT infrastructure.	MUST	Removed. Merged with TR.10 and TR.11.
TR.24	When the Hypermedia MAS generates a new task, an agent within the Hypermedia MAS shall notify the smart contract service (see TR.20). The format of a task should be in JSON matched with smart contract API. The smart contract service shall then notify the smart contract application on the correct computational resource (see TR.46)	MUST	
TR.25	The AI on the edge shall provide information about available workpieces to the Hypermedia MAS. This information shall include at least the following information: An ID of the workpiece. The diameter and centre coordinates of the largest possible circle which can be placed on the workpiece.	MUST	Updated. AI moved from the robot to the edge.

TR.26	The 5G system must provide a MEC service interface, which enables the Edge Orchestrator to make app specific reservations during edge app provisioning and reconfiguration.	MUST	
TR.27	The 5G system must provide a MEC service interface, which enables the Edge Orchestrator to continuously read live metrics and status information from the radio system.	MUST	
TR.28	When an agent in the Hypermedia MAS requires the allocation of an edge app, it shall send a create orchestration request to the Edge Orchestrator. The orchestration request includes the template id derived from the thing description template (see TR.1), received earlier. Further, the create orchestration request can include constraints (e.g., QoS, HW requirements, time, availability, etc.), the type of orchestration (static, dynamic), the targeted edge devices (group, all) and other optimization parameters.	MUST	
TR.29	While an operator is controlling a robot, the robot controller shall provide the following information to the HIL-AR Application framework: - Coordinates of robot to update the digital twin	MUST	Removed. AR Application is directly controlling the UR5 robot.
TR.30	While an operator is controlling a robot, the HIL-AR Application must provide the following information to the robot controller: - Coordinates of digital twin to initiate the robot arm movement - Grab commands - Release commands	MUST	
TR.31	When a new service becomes available, where this service needs to be contacted by an agent (e.g., for a query or a task) during the execution of the use case scenario, the service shall register to the Hypermedia MAS Infrastructure. To register, the service shall provide a service description that adheres to the specification for W3C WoT Thing Descriptions (TDs).	MUST	Removed. Covered by TR.28.
TR.32	If a service shall only be instantiated at run time, a description must be provided for the service that adheres to the specification for W3C WoT Thing Description Templates (TDTs) instead of a Thing Description. However, note that TR.1 needs to be satisfied still.	MUST	Updated to increase clarity of the TD vs. TDT usage.
TR.33	When an agent intends to task a service that is represented through a W3C WoT TDT, the agent contacts the Edge Orchestrator to resolve this TDT to a protocol-bound W3C WoT TD. The EO itself may itself be registered to the Hypermedia MAS Infrastructure.	MUST	Removed. Covered by TR.28.
TR.34	When the AI is deciding how to grab a workpiece, it shall provide a confidence level for its decision.	MUST	Updated. Using predefined coordinates.

TR.35	The Goal Specification Front End may invoke an external service to generate the required output image file format.	MAY	Removed. ASCII text used as input.
TR.36	The back end of the Web-based IDE shall represent the interface to the Hypermedia MAS that is running on the Hypermedia MAS Infrastructure. It provides procedural knowledge as well as organizational knowledge that is configured by a human user to the Hypermedia MAS.	MUST	
TR.37	When a machine accepts a job from an Agent, this Agent shall provide a job description in a format which is understood by the particular machine.	MUST	Removed. Covered by GNFR.18.
TR.38	When a robot is requested to grab a workpiece, AI on the plant edge shall provide the grabbing point and the pose (x, y coordinates and the gripper angle) where the workpiece has to be grabbed.	MUST	Updated. In accordance with the labels of the training data.
TR.39	When a robot is requested to place a workpiece, AI on robot shall provide the place command with the pose (coordinates and orientation of the gripper x,y,z,pitch,yaw,roll) where the workpiece has to be placed.	MUST	Removed. Placing of the workpiece is now pre-programmed
TR.40	When the HIL application requests a robot to move, it shall send the movement command in the form of a UR5 (universal robot) compatible format.	MUST	Updated. The AI and the HIL Application will use different command methods.
TR.40a	When AI requests robot controller to move the robot to a grab spot using x, y, alpha coordinates, robot controller shall translate the coordinates to TCP coordinates a UR5 compatible format.	MUST	New.
TR.40b	When Hypermedia MAS requests robot controller to move the robot to a named position, robot controller shall translate the position to TCP coordinates a UR5 compatible format.	MUST	New.
TR.41	When edge orchestrator requests TSN controller or 5G network controller to setup or modify a (tactile) network connection it shall at least provide the following information: - Source ID (e.g., MAC address or device name) - Destination ID (e.g., MAC address or device name) - Traffic pattern, i.e., Bytes per cycle time[10] (GBR) - Optional Delay constraints - Optional send window[11] constraints - Optional jitter constraints Note: As TSN controller solely controls the wired part of the network and communication resource manager 5G communication manager solely controls the wireless part of the network, edge orchestrator shall manage the split	MUST	Updated. For the ease of use of the interface we allow for devices names instead of MAC addresses. Refactoring of the architecture resulted in the edge orchestrator brokering the service request.

[10] Cycle time refers to the time a cyclic application, e.g. a robot controller, repeats it's control loop, i.e. sends updated information, or to the frequency of frames sent e.g. from a video source.
[11] Send window refers to a time window within the data should be sent, e.g. a window starting 100μs and ending 200μs after cycle start.

	between wired and wireless part of a requested communication service.		
TR.42	The HIL service shall be realized as an edge app.	MUST	Updated. The original wording was confusing.
TR.43	When edge orchestrator requests TSN controller or 5G communication manager to remove a (tactile) network connection it shall at least provide the following information: - Source ID - Destination ID - Session or bearer ID	MUST	Updated according to TR.41.
TR.44	When HIL service requests help from a human operator, it shall publish a help request through MQTT.	MUST	
TR.44a	When an operator wants to take over a help request, it shall send an operator takeover event containing its own IP address to HIL service.	MUST	New
TR.45	When an Agent in the Hypermedia MAS requests for the allocation of edge resources, it shall send a capability description of the required resources to the edge orchestrator. This includes the resource type, so that it can be mapped to edge apps, as well as constraints (QoS, HW requirements, time, availability), which will need to be supplied to the Agent by a suitable service.	SHOULD	Removed. Covered by TR.3.
TR.46	When receiving a notification about a new task, the smart contract service shall notify the smart contract application on the correct computational resource (see TR.24)	MUST	
TR.47	While a production service, e.g., a machine or robot, is registered to the Hypermedia MAS Infrastructure, when one of the events listed below occurs, the machine/robot shall send a state information update to the Hypermedia MAS Infrastructure. Events to be considered are: - The keepalive interval of 30s has expired - A new job was assigned - A job was started - A job was finished - The health state changed - A maintenance request occurred - A maintenance request was cleared - A smart contract concerning the machine was changed	MUST	Removed. Architecture reworked. Switched from REST to polling. See TR.15.
TR.48	When a robot is requested to grab a workpiece, robot controller shall place the camera in a predefined position to enable the AI to capture an image of the workpiece. Predefined positions are required for	MUST	Removed. We now have dedicated cameras for each viewpoint.

	1. the milling machine 2. the laser cutter 3. the storage After reaching the predefined position, robot controller shall trigger AI to capture an image and compute the grab spot pose.		
TR.49	When a robot is requested to pick or place a workpiece, robot controller must compute a pick, respectively place command with a set of four UR5 compatible coordinates from the pose it has received from AI, avoiding collisions with any obstacles. When robot controller is requested to move the robot, robot controller must compute a set of UR5 compatible coordinates from the pose it has received from AI, avoiding collisions with any obstacles.	MUST	Updated. Expanded description.
TR.50	When HIL application requests a robot to move, HIL application shall command the robot using the UR5 API.	MUST	Updated. The HIL Service is only used to establish the connection. The HIL application will use the UR5 APIs.
TR.51	IAKM must register to the MEC infrastructure. Registration includes at least the data usage purpose and its security credential.	MUST	
TR.52	The MEC must register to the IAKM. Registration includes at least MEC security credentials.	MUST	
TR.53	The Global AI should provide information of the AI model, training environments, etc. to the IAKM. This information should use a RDF knowledge graph.	SHOULD	Updated. The Global AI Component replaced all other central servers used in the federated learning implementation, including the Edge AI. Also, the responsibilities of the centralized federated learning component were divided between the Global AI and the IAKM Server. Lastly, the use of RDF knowledge graphs is considered superior to the JSON format by the consortium.
TR.54	All communication between the IAKM and actors must be secured by at least ssl-level security.	MUST	

TR.55	Any MEC module must register to the MEC infrastructure. Registration should include security credentials.	MUST	
TR.56	Any MEC modules should expose its existence and can discover that of other modules. Discovery format should at least include a unique ID, a purpose description and a group ID.	SHOULD	
TR.57	Event captors may be able to monitor telemetry received from the edge devices to detect suspicious activity	MAY	Updated. We may monitor the robot instead of the camera itself.
TR.58	When an event captor has detected malicious activity, it shall send an alert to the Security Assurance Platform	MUST	
TR.59	Given the two different labelled datasets, two Manufacturing AI models are trained to infer the labels given the inputs corresponding to the training data. 1. Grabbing: camera images of workpieces labelled with grabbing location (x, y) coordinates and robot gripper's angle of approach 2. Identifying usable area: images of the workpieces labelled with the usable area (coordinates of the centre and the radius of the circular region), two Manufacturing AI models are trained to infer the labels given the inputs corresponding to the training data.	MUST	Updated. Modified based on the uncertainty of data collection.
TR.60	Pre-training of the Manufacturing AI model uses two different labelled datasets: 1. Grabbing: camera images of workpieces labelled with grabbing location (x, y) coordinates and robot gripper's angle of approach 2. Identifying usable area: images of the workpieces labelled with the usable area (coordinates of the centre and the radius of the circular region) These labelled data is either collected from the demo setup and provided by a human operator or augmented from simulated setting.	MUST	Updated. Added uncertainty of data collection and introduced data augmentation.
TR.61	The Security Assurance platform must be able to aggregate, ingest, and store evidence received from the Event Captors.	MUST	
TR.62	A RabbitMQ message broker must be available and configured as needed for the interaction between Trust enablers.	MUST	
TR.63	The secure AMQP port 5671 must not be blocked by a Firewall	MUST	
TR.64	For the installation and/or operation of the Trust Enablers, root access must be available	MUST	
TR.65	Each edge device must host a version of the MTD Client	MUST	

TR.66	When an MTD Client connects to the MTD server, information must be provided, with at least the following: • Client Public Key or JWT • Edge device IP • Edge device MAC address	MUST	
TR.67	When the MTD server receives a warning, the information must include the following data: • Event Captor / IDS / Assurance Platform Public Key or JWT • Warning type • Additional information based on warning type	MUST	
TR.68	When an MTD client is to disconnect (shutdown), it should inform the MTD server by sending the following information: • Client Public Key or JWT • Disconnect reason (Optional)	SHOULD	
TR.69	When the MTD server sends keepalive (KA) messages to the clients, the clients must answer within a specific time interval by sending the following information: • Client Public Key or JWT • Status If a client fails to comply, it is considered disconnected and its status is logged	MUST	
TR.70	When an MTD Client receives a new configuration, the information must include the following data: • Server Public Key or JWT • Additional information based on the configuration	MUST	
TR.71	Each edge device must host a version of the Trust IDS	MUST	Updated naming.
TR.72	For each device connected to the Interoperability Box, a configuration must be provided that describes the device's properties and available actions.	MUST	
TR.73	For each device connected to the Interoperability Box or set of such devices, an edge device that hosts an Interoperability Box must be provided	MUST	Removed. Redundant to TR.72.
TR.74	Every edge application requiring service level guarantees (e.g., HIL service and HIL application) should provide an interface to allow edge orchestrator to pull its health state.	SHOULD	Removed. For service level guarantees health state based on delay and jitter is sufficient.
TR.75	Each application that needs to use a service must be able to obtain an access token from the AAA using OAuth2/OIDC protocols.	MUST	

TR.76	Each service must be able to validate access tokens through the AAA using OAuth2 protocol to grant access to applications.	MUST	
TR.77	Given the labelled dataset, the obstacle bypassing AI model is trained to infer the labels given the inputs corresponding to the training data. Dataset includes video frames of bypassing obstacles along linear and angular velocity components corresponding to a human driver.	MUST	Updated. Removed data collection and revised based on model training.
TR.78	When pre-training the agriculture AI model, the human operator shall provide the labelled dataset including video frames of bypassing obstacles along linear and angular velocity components corresponding to a human driver.	MUST	
TR.79	Edge apps must be orchestrated with docker-compose as one or multiple docker containers.	MUST	New
TR.80	Edge apps requiring to exposing ports to outside on the host must be able to set the port number within the port range [32768, 60999]	MUST	New
TR.81	Edge App container port numbers must be at least 8000	MUST	New
TR.82	Edge apps must be x86 compatible and Linux-based	MUST	New
TR.83	Edge apps must be ready to be stopped and started at any time.	MUST	New
TR.84	Edge apps must be able to self-reliantly recreate their state after start (local state might be destroyed after each shutdown)	MUST	New
TR.85	Edge app developer must comply to the guidelines for the Hello Edge App (https://gitlab.eurecom.fr/intelliot-project/edge/apps/spec-iedge/hello-edge-app)	MUST	New
TR.86	The configuration files for provisioning of Edge Apps have to be available in the edge app's corresponding GitLab repo	MUST	New
TR.87	Edge Apps must not be depending on mounted volumes on the host file systems	MUST	New
TR.88	Edge Apps must not depend on special hardware equipment on the edge device (dongles, GPUs)	MUST	New
TR.89	Edge apps must not expose ports on number 80, 443, 1883,	MUST	New
TR.90	Edge apps must not rely on Internet uplink	MUST	New
TR.91	Edge app must not rely on the environment where they are executed	MUST	New
TR.92	Edge apps are head-less (no GUI)	SHOULD	New
TR.93	Images for edge app container must be available in an accessible container registry	MUST	New
TR.94	Images to be engraved have to be in the SVG format. All objects in the image have to be converted to lines. (Limited set of supported primitives)	MUST	New

TR.95	Text engraving requests must include the text to be engraved (single line, no formatting), the font size (or alternatively the text width and/or the text height), the font name, the north western corner of the text	MUST	New
TR.96	Text engraving requests should include a maximal bounding box with x, y, width and height coordinates	SHOULD	New
Open Call 1 -related Technical Requirements			
OC1.TR.1	The Trilogis local server shall collect, filter, and dispatch (through web API) the location information acquired by the RFID readers about the tagged workpieces moving throughout the proximity areas	MUST	New
OC1.TR.2	The Trilogis software platform shall visualize the values of the Key Performance Indicators (KPIs) through interactive graphs	MUST	New
OC1.TR.3	The graphs showing the values of the KPIs should be configurable. For example, the user should be able to select the time window of the values	SHOULD	New
OC1.TR.4	The typology and the number of the KPIs reported in the dashboard should be selectable by the end user	SHOULD	New
OC1.TR.5	The position of the tagged workpieces in the defined proximity areas should be visible on a map updated in real-time	SHOULD	New
OC1.TR.6	The entrances and the exits of all the tagged workpieces from the proximity areas should be notified in real-time	SHOULD	New
OC1.TR.7	The proximity areas may be editable (size, shape, position) by the end user through a dedicated GUI	MAY	New
OC1.TR.8	Each proximity area shall be monitored by a RFID reader. The number of readers is equal to the number of predefined areas	MUST	New
OC1.TR.9	Workpieces need to be tagged with RFID tags. The ID of each RFID tag is unique	MUST	New
OC1.TR.10	The size of the detection area is predefined and static. The coverage and the position of each RFID reader have to be calibrated accordingly	MUST	New
OC1.TR.11	The layout and geolocation of the setup is required to define the size and the position of the proximity areas	MUST	New
OC1.TR.12	The RFID readers need to be connected to internet for data transmission to the remote server	MUST	New
OC1.TR.13	Remote connection to the RFID controllers for system management and maintenance is required	MUST	New
OC1.TR.14	The software platform TRACCIA by Trilogis shall be accessible as a web application using a browser and the user credentials provided by administrators	MUST	New
OC1.TR.15	Access to the Hypermedia MAS APIs should be granted to TRACCIA backend in order to compute the KPIs	MUST	New

OC1.TR.16	One workpiece should not be processed multiple times, in order to compute properly the KPIs related to the process performance	SHOULD	New
OC1.TR.17	The MYW Edge AI Designer (EAD) shall enable the design and delivery of edge intelligence to any equipment, machine, actuator, and sensor.	MUST	New
OC1.TR.18	The MYW Edge AI Designer (EAD) shall support the design, development, training and deployment of ML/AI models.	MUST	New
OC1.TR.19	The MYW Edge AI Designer (EAD) shall offer access to structured and standardized system and sensors information through a usable front end.	MUST	New
OC1.TR.20	The MYW Edge AI Designer (EAD) may support the retrieval and multi-view presentation of both real time and historical data.	MAY	New
OC1.TR.21	The MYW Real-Time Analyzer (RTA) shall be able to notify anomalies, alarms, and relevant events to trigger the human intervention (HIL).	MUST	New
OC1.TR.22	The MYW Real-Time Analyzer (RTA) shall support supervised/unsupervised clustering algorithms to data streams acquired for a significant amount of time.	MUST	New
OC1.TR.23	The Real-Time Analyzer (RTA) should iteratively and incrementally refine/update existing ML models and AI algorithms by applying retraining methods to new incoming data streams.	SHOULD	New
OC1.TR.24	The MYW Real-Time Analyzer (RTA) shall manage the deployment of developed and trained AI algorithms in the execution environment on CPU empowered edge/very edge processing units.	MUST	New
OC1.TR.25	The MYW Edge Processing Unit (EPU) shall be able to run AI algorithms on the local CPU.	MUST	New
OC1.TR.26	The MYW Edge Processing Unit (EPU) may be able to run AI algorithms on the local FPGA as well as on dedicated chip-based neural networks (ASIC).	MAY	New
OC1.TR.27	The MYW Edge Processing Unit (EPU) shall manage the collection and ingestion of data acquired and transmitted by heterogeneous sensors, timestamping and synchronizing them.	MUST	New
OC1.TR.28	The MYW Edge Processing Unit (EPU) shall perform the pre-processing of data from sensors before their analysis and interpretation/classification.	MUST	New
OC1.TR.29	The MYW Edge Processing Unit (EPU) should support a set of suitable methods for data preparation and features extraction.	SHOULD	New
OC1.TR.30	The MYW Edge Processing Unit (EPU) shall perform the real-time detection of anomalies and deviations in the data	MUST	New

	streams acquired and transmitted by heterogeneous sensors, according to the available local AI models.		
OC1.TR.31	The MYW Smart Tag shall be equipped with an Inertial Measurement Unit (IMU), a combination of an accelerometer and a gyroscope to detect movements and measure the intensity of movements in terms of acceleration and rotational speeds.	MUST	New
OC1.TR.32	The MYW Smart Tag shall offer connectivity through both BLE 5.1 (Bluetooth Low Energy) and WiFi communication protocols.	MUST	New
OC1.TR.33	The MYW Tracking Device should be equipped with an Inertial Measurement Unit (IMU) to detect movements and measure the intensity of movements in terms of acceleration and rotational speeds.	SHOULD	New
OC1.TR.34	The MYW Tracking Device may be equipped with sensors to measure environmental/atmospheric parameters (temperature, humidity, pressure, etc.).	MAY	New
OC1.TR.35	The MYW Tracking Device should offer connectivity through both WiFi and cellular (3G/4G) communication protocols.	SHOULD	New
OC1.TR.36	Patients should have Google accounts to use Google fit integration	SHOULD	New
OC1.TR.37	Patients' mobile applications should have internet connectivity (WiFi and cellular or (4G/5G) to have real time feedback from doctors'	MUST	New
OC1.TR.38	Patients should use the oximeter medical device while performing physical activity tasks related to their careplan	MUST	New
OC1.TR.39	Patients' smartphone should be paired with their medical devices through Android interface before first use	MUST	New
OC1.TR.40	IKH's robot must connect to the network either though WIFI or 5G connection	MUST	New
OC1.TR.41	Hypermedia MAS must send commands in continuous mater	MUST	New
OC1.TR.42	IKH's ROS bridge to Hypermedia MAS and Unity must be able to write on ROS topics	MUST	New
OC1.TR.43	IKH's ROS bridge must receive web sockets API from external sources (e.g., Unity or Hypermedia MAS)	MUST	New
OC1.TR.44	IKH's farmer's logbook must log its event through the DLT manager	MUST	New
OC1.TR.45	IKH's farmer's logbook should support logging events coming from automated sources	SHOULD	New
OC1.TR.46	IKH's VR application must control a remote moving platform and a robotic arm using the 2 controllers of Oculus Quest2	MUST	New

4 ALIGNMENT WITH PROJECT OBJECTIVES & REQUIREMENTS

The components comprising the IntellIoT architecture have been conceived and are being designed and developed with the project's overarching objectives in mind. Furthermore, refinements to these components, stemming from the project's research and development activities, are also driven by IntellIoT's requirements, as defined within Deliverable D2.5 ("Technology Analysis & Requirements Specification (final version)"). This section aims to present the current status in terms of coverage of project objectives and requirements (as defined in D2.5 and in Section 3 above) through the various components comprising IntellIoT. Said preliminary mapping presented herein will also assist in developing the components, while also guiding the requirements update that is to follow in Cycle 2 of the project. The updated, final list of components and their mapping to the (also updated) final project requirements will be presented in the follow-up deliverable, D2.6 - "High level architecture (final version)".

4.1 Mapping to Objectives

Table 12 aims to present the role of each individual component in relation to the project's overarching technical objectives (Objectives 1-5).

Table 12. Mapping of IntellIoT's objectives to pertinent architectural building blocks (components)

	Component	Obj. 1	Obj. 2	Obj. 3	Obj. 4	Obj. 5
Collaborative IoT	Global AI	X		X		X
	Local AI	X		X		X
	Hypermedia MAS Infrastructure	X				X
	Web-based IDE for Hypermedia MAS	X				X
	IAKM Server	X		X		X
	IAKM Client	X		X		X
	Interoperability Box	X				X
HIL Enablers	Data Repository			X		X
	HIL Service	X		X		X
	HMIs	X		X		X
	HIL Application	X		X		X
Trust Enablers	Security Assurance Platform				X	X
	Event Captors				X	X
	AAA				X	X
	Intrusion Detection System				X	X
	Moving Target Defenses				X	X
	Resource-aware DLT Manager				X	X
Infrastructure Management	Edge Orchestrator	X				X
	Edge Manager	X				X
	Computational Resource Manager	X				X
	TSN Controller		X			X
	Communication Resource Manager		X			X
	5G Low Latency MEC		X			X
	Private 5G Core		X			X

	5G RAN	X	X
	5G RAN Controller	X	X
	5G Network Monitoring	X	X

4.2 Mapping to Requirements

The tables that follow provide a mapping of the components comprising the IntellIoT framework to the project's General Functional (Table 13), Use Case-specific Functional (Agriculture, Healthcare and Manufacturing, in Table 14, Table 15, and Table 16, respectively), General Non-Functional (Table 17), and Use Case-specific Non-functional (Table 18), as these have been defined in deliverable D2.5. Moreover, a mapping to the Technical Requirements, as defined in Section 3 above, is provided in Table 19.

Table 13. Mapping of IntellIoT components to General Functional Requirements (see D2.5)

	Component	GFR.1	GFR.3	GFR.4	GFR.5	GFR.6	GFR.7	GFR.10	GFR.11	GFR.12	GFR.13	GFR.14	GFR.15	GFR.16	GFR.17	GFR.18	GFR.19	GFR.20	GFR.21	GFR.22
Collaborative IoT	Global AI								X											
	Local AI								X											
	Interoperability BoX																			
	Hypermedia MAS Infrastructure	X	X	X				X												
	IAKM Server																		X	
	IAKM Client																		X	
HIL	HIL Service									X	X	X		X			X			
	HIL Application				X					X	X	X	X	X	X	X	X	X		
	HMIs												X	X			X	X		
	Data Repository																			
Trust	Security Assurance Platform																			X
	Event Captors																			
	AAA																			
	Intrusion Detection System Instance																			
	Moving Target Defenses																			
	Resource-aware DLT Manager		X		X	X	X													
Infrastructure Management	Edge Orchestrator																			
	Edge Manager																			
	Computational Resource Manager																			
	TSN Controller									X										
	Communication Resource Manager																			

Table 14. Mapping of IntelIIoT components to Agriculture UC-specific Functional Requirements (see D2.5)

	Components	FR. UC1 #1	FR. UC1 #2	FR. UC1 #3	FR. UC1 #5	FR. UC1 #7	FR. UC1 #8	FR. UC1 #10	FR. UC1 #11	FR. UC1 #12	FR. UC1 #13	FR. UC1 #14	FR. UC1 #15	FR. UC1 #17	FR. UC1 #18	FR. UC1 #19	FR. UC1 #20
Collaborative IoT	Local AI						X	X									
	Global AI						X	X									
	Hypermedia MAS Infrastructure		X														
	Web-based IDE for Hypermedia MAS																
	IAKM Server						X										
	IAKM Client						X										
	Interoperability Box																
HIL	Data Repository																
	HIL Application								X	X	X	X	X	X	X	X	X
	HMI: VR Glasses (Oculus Quest 2) & game controller								X	X	X	X	X	X	X	X	
	HIL Service												X				
Trust	Security Assurance Platform																
	Event Captors																
	AAA																
	Intrusion Detection System																
	Moving Target Defenses																
	Resource-aware DLT Manager			X													
Infrastructure Management	Edge Orchestrator		X														
	Edge Manager		X														
	Communication Resource Manager									X							
	5G LL MEC									X					X		
	Private 5G Core									X					X		
	5G RAN									X					X		
	5G RAN Controller									X					X		
	5G Network Monitoring									X					X		
UC	Edge Device Runtime('Tractor Contoller')			X	X	X	X										

Table 15. Mapping of IntelIIoT components to Healthcare UC-specific Functional Requirements (see D2.5)

	Component	FR. UC2 #1	FR. UC2 #2	FR. UC2 #3	FR. UC2 #4	FR. UC2 #5	FR. UC2 #6	FR. UC2 #7	FR. UC2 #8	FR. UC2 #9	FR. UC2 #10	FR. UC2 #11	FR. UC2 #12	FR. UC2 #13	FR. UC2 #14	FR. UC2 #15	FR. UC2 #16
Collaborative IoT	Global AI		X						X					X	X	X	X
	Local AI		X					X	X					X	X	X	X
	Interoperability Box	X			X		X										
	Hypermedia MAS Infrastructure	X			X		X										
	Web-based IDE for Hypermedia MAS	X			X		X										
	IAKM Client								X					X	X	X	X
	IAKM Server								X					X	X	X	X
HIL	Healthcare AI Models		X			X			X					X	X	X	X
	HMIs							X	X			X	X				
	Data Repository		X	X		X		X	X				X				
Trust	Security Assurance Platform									X	X						
	Event Captor										X						
	Resource-aware DLT Manager									X	X						
Infra. Mngmnt	5G LL MEC		X					X	X			X	X	X	X	X	X
	5G Network Monitoring		X					X	X			X	X	X	X	X	X
UC	Smart Devices	X	X	X	X	X	X							X	X		

Table 16. Mapping of IntellIoT components to Manufacturing UC-specific Functional Requirements (see D2.5)

	Component	FR. UC3 #1	FR. UC3 #2	FR. UC3 #3	FR. UC3 #4	FR. UC3 #5	FR. UC3 #6	FR. UC3 #7	FR. UC3 #8
Collab. IoT	Hypermedia MAS Infrastructure	X		X					
	Web-based IDE for Hypermedia MAS			X					
HIL	HIL Application				X	X	X		
	AR Glasses (Hololens 2)				X		X		
Trust	Resource-aware DLT Manager								X
Infrastructure Management	Edge Orchestrator		X						
	Edge Manager		X						
	Computational Resource Manager		X						
	TSN Controller		X						
	Communication Resource Manager				X				
	5G LL MEC		X		X	X	X		
	Private 5G Core				X	X	X		
	5G RAN				X	X	X		
	5G RAN Controller				X	X	X		
	5G Network Monitoring				X	X	X		

Table 17. Mapping of IntellIoT components to General Non-functional Requirements (see D2.5)

	Component	GNFR. 1	GNFR. 2	GNFR. 3	GNFR. 4	GNFR. 5	GNFR. 6	GNFR. 7	GNFR. 8	GNFR. 9	GNFR. 10	GNFR. 11	GNFR. 12	GNFR. 13	GNFR. 14	GNFR. 15	GNFR. 16	GNFR. 17	GNFR. 18
Collab. IoT	Hypermedia MAS Infrastructure													X			X	X	X
	Web-based IDE for Hypermedia MAS													X			X	X	X
	IAKM Client														X				
	IAKM Server														X				
HIL	Data Repository																		
	HIL Application								X	X			X	X		X			
	HMIs								X	X						X			
	HIL Service																		
Trust	Security Assurance Platform	X	X	X		X	X	X	X	X	X								
	Event Captor	X	X																
	AAA									X									
	Intrusion Detection System			X	X														
	Moving Target Defenses			X		X						X							
	TSN Controller															X			
	Resource-aware DLT Manager					X	X	X	X										
Infrastructure Management	TSN Controller															X			
	Communication Resource Manager															X			
	5G LL MEC															X			
	Private 5G Core															X			
	5G RAN															X			
	5G RAN Controller															X			
	5G Network Monitoring															X			

Table 18. Mapping of IntellIoT components to UC-specific Non-functional Requirements (see D2.5)

	Component	UCNFR.2	UCNFR.3	UCNFR.4	UCNFR.5	UCNFR.6	UCNFR.7	UCNFR.9
Collab. IoT	Hypermedia MAS Infrastructure						X	
	Interoperability Box							
HIL	Data Repository			X				
	HIL Application	X						
	HMIs	X		X				
	HIL Service		X					
Trust	DLT Manager	X	X					
	DLT Client	X	X					
Infrastructure Management	Edge Device Runtime	X	X					
	Communication Resource Manager							X
	5G LL MEC							
	Private 5G Core							
	5G RAN							
	5G RAN Controller							
	5G Network Monitoring							

Table 19. Mapping of IntelllioT components to Technical Requirements (TR.1-50 top, TR.51-96 bottom; see Section 3 above)

	Component	TR.1	TR.2	TR.3	TR.4	TR.5	TR.6	TR.7	TR.8	TR.9	TR.10	TR.11	TR.12	TR.13	TR.14	TR.15	TR.16	TR.17	TR.18	TR.19	TR.20	TR.21	TR.22	TR.23	TR.24	TR.25	TR.26	TR.27	TR.28	TR.29	TR.30	TR.31	TR.32	TR.33	TR.34	TR.35	TR.36	TR.37	TR.38	TR.39	TR.40	TR.41	TR.42	TR.43	TR.44	TR.45	TR.46	TR.47	TR.48	TR.49	TR.50
Collab. IoT	Global AI									X													X			X									X				X	X	X									
	Local AI									X													X			X									X				X	X	X									
	Hypermedia MAS Infrastructure	X	X	(X)	(X)	X	X	(X)					X		X	(X)	(X)	(X)					(X)		X	(X)			X			(X)		X				X							X		(X)				
	Web-based IDE for Hypermedia MAS																																			X															
	IAKM Client								X																																											
	IAKM Server								X																																											
	Interoperability Box																																																			
HIL	HIL Service																																										X	X	X	X					X
	System data Repository																																																			
	HIL Application																						X							X	X										X				X					X
	HMIs																													X	X					X					X								X
Trust	Security Assurance Platform																																																		
	Event Captor																																																			
	AAA																					X																															
	Intrusion Detection System																																																		
	Moving Target Defenses																																																		
	TSN Controller																																											X		X						
	Resource-aware DLT Manager																				X			X																								X					
Infrastructure Management	Edge Orchestrator			X																						X	X	X					X									X								
	Edge Manager			X																						X																									
	Computational Resource Manager			X																						X		X																							
	TSN Controller																																										X		X						
	Communication Resource Manager																																										X		X						
	5G LL MEC																										X	X																X		X						
	Private 5G Core																																											X		X						
	5G RAN																																										X		X						
	5G NFV Controller																																										X		X						
	5G RAN Controller																																										X		X						
	5G Network Monitoring																																											X		X						
UC	Edge device										X	X		X					X	X														X													X	X		

	Component	TR.51	TR.52	TR.53	TR.54	TR.55	TR.56	TR.57	TR.58	TR.59	TR.60	TR.61	TR.62	TR.63	TR.64	TR.65	TR.66	TR.67	TR.68	TR.69	TR.70	TR.71	TR.72	TR.73	TR.74	TR.75	TR.76	TR.77	TR.78	TR.79	TR.80	TR.81	TR.82	TR.83	TR.84	TR.85	TR.86	TR.87	TR.88	TR.89	TR.90	TR.91	TR.92	TR.93	TR.94	TR.95	TR.96
Collab. IoT	Global AI			X																																										
	Local AI			X																																										
	Hypermedia MAS Infrastructure																																													
	Web-based IDE for Hypermedia MAS																																													
	IAKM Client				X																																									
	IAKM Server				X																																									
	Interoperability Box																						X	X																							
HIL	HIL Service									X	X														X			X	X																	
	System data Repository									X	X																	X	X																	
	HIL Application																								X																					
	HMIs																																													
Trust	Security Assurance Platform								X			X	X	X																																
	Event Captor							X	X			X																																			
	AAA																									X	X																			
	Intrusion Detection System																					X																								
	Moving Target Defenses															X	X	X	X	X	X																									
	TSN Controller																																													
	Resource-aware DLT Manager																																													
Infrastructure Management	Edge Orchestrator	X				X	X																		X																					
	Edge Manager																								X																					
	Computational Resource Manager																																														
	TSN Controller																																													
	Communication Resource Manager																													X	X	X	X			X	X	X	X	X	X	X	X	X			
	5G LL MEC	X	X			X	X																																							
	Private 5G Core																																													
	5G RAN																																													
	5G NFV Controller																																													
	5G RAN Controller																																													
	5G Network Monitoring																																													
UC	Edge device														X	X																															

5 RELATION TO OTHER IOT ARCHITECTURES

There is a plethora of efforts aiming to define a "Reference" IoT architecture, motivated by the need to alleviate the interoperability barriers to IoT adoption, provide a common vocabulary to IoT stakeholders and, in general, allow the IoT to realise its full potential. Some prominent efforts in this regard include: the World Wide Web Consortium's (W3C) Web of Things (WoT)[12] Architecture [4]; the European Edge Computing Consortium's (EECC)[13] Reference Architecture Model Edge Computing (RAMEC); the Alliance for Internet of Things Innovation (AIOTI)[14] with its IoT "High Level Architecture (HLA)"; the Industrial Internet Consortium's (IIC) "Industrial Internet Reference Architecture (IIRA)" [6]; the International Telecommunication Union, Telecommunication Standardization Sector's (ITU-T) – "IoT Reference Model" included within Recommendation Y.4000 /Y.2060 [7]; the Big Data Value Association's (BDVA) "Big Data Value Reference Model" (included in [8]), and; the oneM2M's [15] "Functional Architecture" [9]. These are augmented by other efforts aiming to provide structured ways to view and manage new developments in the IoT landscape, such as the move to Edge Computing, e.g., the Edge Taxonomy proposed by IDC [10] (and elaborated upon by the E.C.).

While from the perspective of the project it was important to define an architecture that accurately reflects the project's concept and, thus, prominently features the three pillars of IntellIoT (i.e., Collaborative IoT, Human-in-the-loop and Trustworthiness), the project's work is not detached from the rest of the IoT and NGIoT landscape. In fact, IntellIoT's architecture can be mapped to and is compatible with the most prominent efforts to define a reference IoT architecture.

The subsections that follow showcase this, studying the alignment of IntellIoT's architecture with such reference architectures that are well-established, current (i.e., maintained) and relevant to the project's scope and activities. For this, an abstracted, high-level view of IntellIoT's logical architecture is considered (to facilitate readability of the figures), which is then mapped to the corresponding diagrams of the examined architectures, while details on the rationale of the mapping are provided in the accompanying text.

5.1 W3C Web of Things (WoT) Architecture

The purpose of this section is to explain how the architectural components of IntellIoT, and more specifically of IntellIoT's Collaborative IoT aspect, align with the W3C Web of Things Architecture [4]. First, we give a concise overview of the relevant aspects of the W3C WoT Architecture. Then, we discuss the alignment of relevant components of IntellIoT with the W3C WoT Architecture.

5.1.1 ARCHITECTURAL ELEMENTS OF THE W3C WOT ARCHITECTURE

The two core elements of the W3C WoT Architecture (an abstract view shown in Figure 49) are **Web Things ("Things")** that are used by **Consumers**. Things are abstract physical (e.g., an industrial robot) or virtual (e.g., an orchestration service) entities and must be described with **W3C WoT Thing Descriptions** (TDs). Consumers must be able to parse and process the JSON-based TD representation format - classic JSON or RDF serializations such as Turtle or JSON-LD, where the use of RDF enables the semantic processing of the Thing metadata (e.g., enabling semantic inference). TDs, then, are the default external textual Web representation of a Thing (TD is therefore often referred to as "HTML for Smart Things"). Importantly, TDs are instance-specific (i.e., they describe a specific Thing rather than types of Things).

[12] https://www.w3.org/WoT/
[13] https://ecconsortium.eu/
[14] https://aioti.eu/
[15] https://www.onem2m.org/

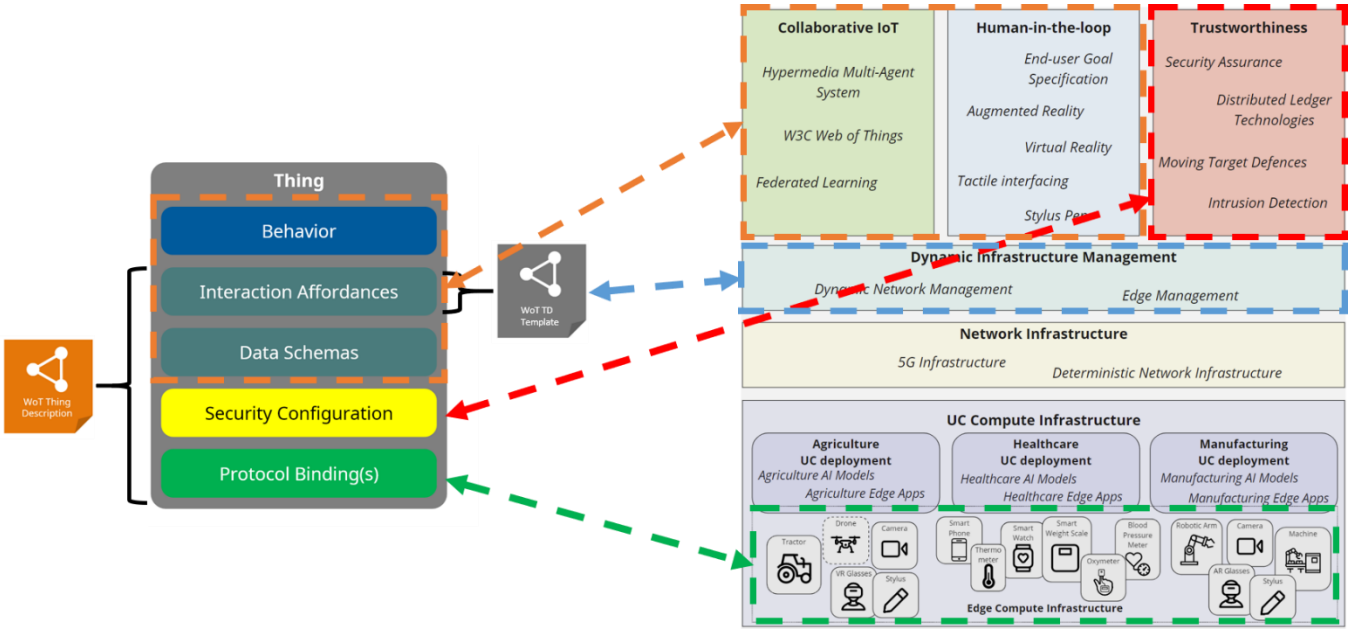

Figure 49.Abstract architectural alignment with W3C WoT

Next, the W3C WoT Architecture defines so-called **Interaction Affordances** as metadata of a Thing that shows and describes to Consumers in what ways they may interact with the Thing. These Interaction Affordances form an important part of the TD of any Thing and are coupled with one or multiple **Protocol Bindings** that map the Interaction Affordance to concrete messages of a specific protocol such as HTTP(S) or CoAP(S). A general and well-known interaction affordance is "navigation" which is followed by simply dereferencing a Web link; the W3C WoT Architecture in addition defines **Property Affordances** (which expose a state of the Thing, e.g., reading a sensor value), **Action Affordances** (which allow to invoke a function on the Thing, e.g., toggling a lamp), and **Event Affordances** (which allow to asynchronously receive events from the Thing).

Thing Description Templates (TDTs) provide purely logical descriptions of the interface and possible interactions with a device or service, but do not contain device-specific information, such as Protocol Bindings. Finally, W3C WoT defines a security vocabulary that enables the specification of several well-established security mechanisms (**Security Configurations**) within TD Protocol Bindings.

In a **typical interaction scenario**, a Consumer discovers a Thing's TD, parses it to learn about the Thing's Interaction Affordances, selects an affordance to follow, parses the Protocol Binding of that affordance, and follows the affordance by sending messages as specified by the binding. For more details about the W3C WoT Architecture, we defer the user to the published specification (see [4], which also includes an explanation of the terminology).

5.1.2 ALIGNMENT OF INTELLIOT'S COLLABORATIVE IOT COMPONENTS AND THE W3C WOT ARCHITECTURE

In IntellIoT, all services that might be used by consumers in IntellIoT are encapsulated within Edge Apps – in alignment with the W3C WoT Architecture, **these Edge Apps therefore are modeled as W3C WoT Web Things and provide W3C WoT TDs**. Their functionality is advertised in the form of Interaction Affordances within these W3C WoT TDs. For clarification, in IntellIoT, the Hypermedia MAS Infrastructure itself is modelled by means of W3C WoT Web Things as well: it provides affordances that allow consumers to create Agent environments and workspaces and to add other W3C WoT Web Things to workspaces. Furthermore, the main service consumers in IntellIoT are Agents that live in the Hypermedia MAS Infrastructure. From the perspective of the W3C WoT, these Agents therefore are W3C WoT

Consumers. If an Edge App is not deployed, Agents can still be configured to use the functionality of this Edge App at run time. To enable this and in a further alignment step, IntellIoT makes use of **W3C WoT TDTs:** Concretely, in case the W3C WoT TDT at run time still does not provide a protocol binding, the Agent calls upon the **Edge Orchestrator** (itself a W3C WoT Thing) to provision the Edge App (i.e., the W3C WoT Web Thing) and deliver the protocol-bound W3C WoT TD. IntellIoT hence can make **full use of TD-specified Protocol Bindings** as well as **Security Mechanisms** as defined in the W3C WoT Architecture.

Leaving the scope of immediate Consumer-Web Thing-Interaction, in IntellIoT, interactions between W3C WoT Consumers and W3C WoT Web Things are **journaled using a distributed ledger**. Since the interactions between W3C WoT Consumers and W3C WoT Web Things are derived from W3C WoT TDs, the logs in this ledger are **compatible with W3C WoT TDs** as well.

5.2 AIOTI High Level Architecture (HLA)

AIOTI is an initiative established in 2016, aiming to contribute to the creation of a dynamic European IoT ecosystem and speed up the take up of IoT, by driving business, policy, research and innovation development in the IoT & Edge Computing and other converging technologies across the Digital Value Chain. Its members include key European IoT stakeholders (large companies, SMEs and start-ups, research centres, universities, associations, and end-user representatives), with the common goal to support digitization in Europe, and competitiveness of Europe.

A key output of the Standardisation Working Group (WG) of AIOTI is the "High Level Architecture (HLA)", at Release 5.0, since December 2020 [1]. The HLA follows the evolution on IoT Architectural aspects and available specifications, considering existing SDOs and alliances architecture specifications, and aiming to promote architectural convergence with SDOs, alliances, consortia, and other relevant parties (e.g., open-source projects, policy makers, regulators, pilots and test beds, research organizations, companies). Thus, the HLA intentionally remains high-level, to not compete with established SDOs, alliances and open-source projects.

The AIOTI HLA Functional Model, as shown in Figure 50, describes functions and interfaces between functions of an IoT system. It follows a three-layer approach, featuring an **Application layer** (encompassing the communications and interface methods used in process-to-process communications), an **IoT layer** (groups IoT-specific functions, such as data storage and sharing, exposing them to the application layer via APIs), and a **Network layer** (including data plane services, providing short and long range connectivity and data forwarding between entities, and control plane services such as location, device triggering, QoS or determinism).

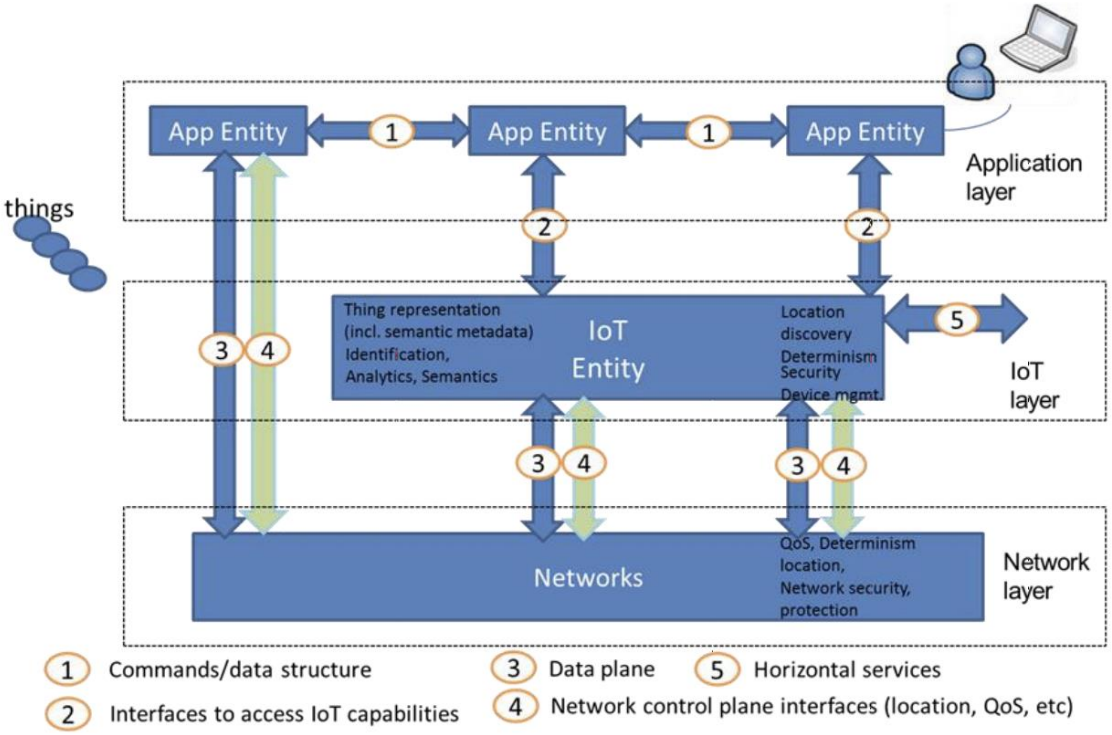

Figure 50. The AIOTI HLA Functional Model

As shown in Figure 50, the HLA functions include:

- **App Entity:** Referring to entities at the application layer that implement IoT application logic (e.g., fleet tracking or health monitoring application). Such an entity can reside on edge devices and/or gateways and/or servers (thus, a centralised approach should not be assumed, per AIOTI's architectural perspective).
- **IoT Entity:** Referring to an entity that exposes IoT functions to App Entities or other IoT entities by the corresponding interfaces and leveraging the underlying Networks (at Network Layer). IoT functions from HLA's perspective can include, for example, data storage and sharing, subscription and notification, device maintenance/firmware upgrade, access rights' management, localisation, analytics, and semantic discovery.
- **Networks:** Referring to the underlying networking (data and control plane) capabilities, via different technologies (PAN, LAN, WAN, etc.), and spanning different administrative network domains. These capabilities are tailored to the App Entities' requirements, including aspects such as QoS, determinism, and security.

It should be highlighted that these functions do not refer to a specific implementation or deployment and, thus, a one-to-one mapping of a function with a physical entity in the operational environment should not be assumed.

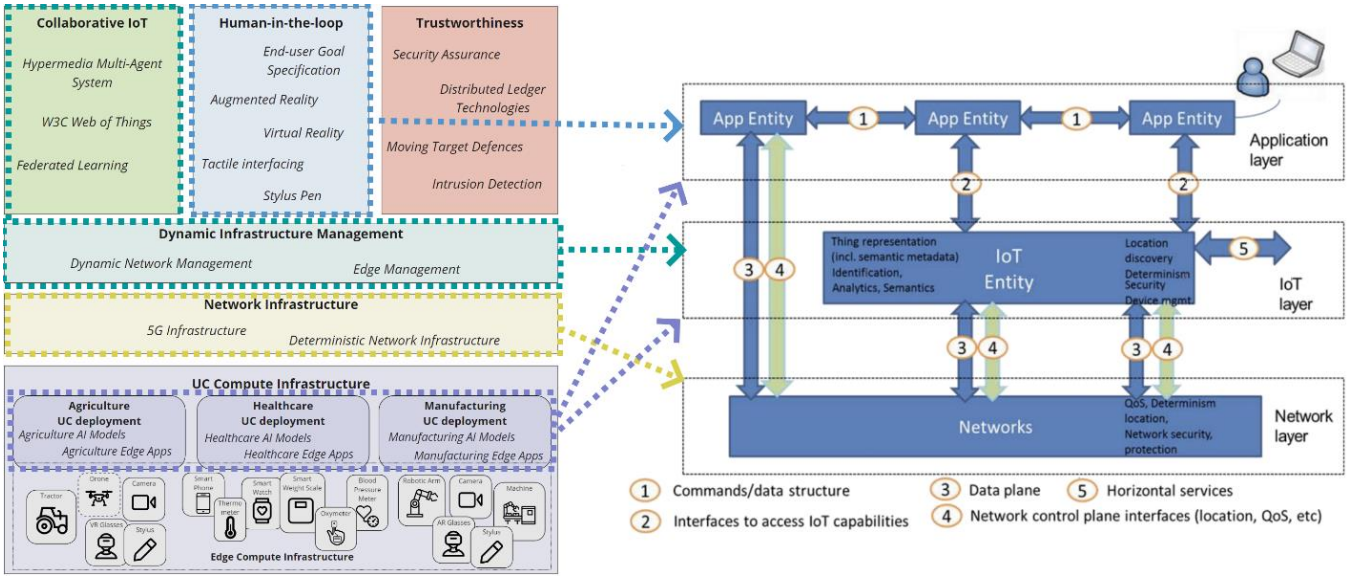

Figure 51. IntellIoT's high level architecture (left) mapped to AIOTI's HLA functional model (right)

Considering the above, a mapping of IntellIoT's high-level architecture to AIOTI's HLA is provided in Figure 51. In more detail:

- The Human-in-the-loop enablers which provide user interfacing capabilities and integrate the corresponding application logic are mapped to the App Entities of the HLA.
- The Trustworthiness enablers can be mapped to all three types of HLA functions, as per AIOTI's approach each of these may embed corresponding security and privacy -related features (e.g., network security mechanisms at the Networks, access control management at the IoT Entities). To facilitate the readability of Figure 51, the arrows from Trustworthiness to all three types of HLA entities are omitted.
- The Collaborative IoT and Dynamic Infrastructure Management enablers are then mapped to IoT Entities, as they provide supporting functions (e.g., semantics, management, analytics) that HLA considers inherent capabilities of said Entities.
- IntellIoT's networking infrastructure, with its data and control plane features, is directly mapped to the Networks of the HLA, and.
- IntellIoT's UC compute infrastructure is mapped to both the App Entities (considering the Edge Apps that are deployed on the UC infrastructure, such as on gateways and other edge devices), and to IoT Entities (considering the Edge devices' capabilities deployed support discovery of things by App Entities and to enable related services such as actuation or measurements, per the HLA model).

In general, the open and high-level approach to the definition of AIOTI's HLA - purposefully chosen to support interoperability and alignment with other architecture standards and initiatives - allows for a straightforward mapping to IntellIoT's architecture. In fact, for a mapping of AIOTI's HLA to additional architectures (such as the oneM2M, ITU-T and ICC architectures mentioned above), we defer the reader to the relevant AIOTI HLA deliverable [5].

5.3 IIC Industrial Internet Reference Architecture (IIRA)

The IIC is an open membership organization founded in 2014 to bring industry stakeholders together to deliver transformative business value to industry, organizations, and society by accelerating adoption of a trustworthy

Internet of Things. To achieve its goals, the IIC strives to create an ecosystem built around insight and thought leadership, promotion of interoperability and security via reference architectures, security frameworks, open standards, and best practice frameworks. They also push for real-world implementations (testbeds) to validate technologies and drive innovation. The core verticals covered by the IIC include IT, Networks, Academia & Research, Manufacturing, Energy & Utilities, and Healthcare. We provide best-practice frameworks and liaisons with Standards Development Organizations.

A key output of the IIC - and the Architecture Task Group, under the Technology Working Group, more specifically - is the IIRA [6] (at Version 1.9 since June 2019); a standards-based open architecture for Industrial Internet of Things (IIoT) systems. The IIRA provides an architectural template and an associated methodology enabling IIoT system architects to design their own systems based on a common framework and concepts. The IIRA description and representation are generic and kept at a high abstraction level to maximize its value (allowing for broad industry applicability, driving interoperability, facilitating mapping to applicable technologies, and guiding technology and standards' development).

The IIRA follows the viewpoint definition approach at the core of the ISO/IEC/IEEE 42010:2011 [11] standard. Viewpoints comprise conventions framing the description and analysis of specific topics of interest ("concerns") pertaining to the system. Within the IIRA four viewpoints are defined: **Business Viewpoint** (focusing on concerns regarding the identification of stakeholders and their business vision, values, and objectives); **Usage Viewpoint** (focusing on the concerns of expected system usage, such as the sequences of activities of involved entities that deliver the intended system capabilities); **Functional Viewpoint** (focusing on concerns regarding the functional components in an IIoT system, including their structure and interrelation, interfaces, interactions, relation and interactions of the system with external elements in the environment), and; **Implementation Viewpoint** (focusing on concerns of the technical representation of an IIoT system, including technologies and components required for implementing the activities and functions prescribed by the usage and functional viewpoints).

For the task of mapping IIRA to the IntellIoT architecture, the Functional Viewpoint is the most relevant choice. The Functional Viewpoint in IIRA decomposes a typical IIoT system into five functional domains, each consisting of building blocks that provide a specific functionality, with data and control flows between the domains. These functional domains are:

- **Control domain:** Represents the collection of functions performed by industrial control and automation systems (e.g., sensing and actuating, control systems in autonomous systems and robotic machines in the manufacturing floor).
- **Operations domain:** Represents the collection of functions responsible for provisioning, management, monitoring and optimization of the systems in the control domain.
- **Information domain**: Represents the collection of functions responsible for aggregating data from the other domains (especially the control domain), and processing them (transforming, persisting, and modelling or analysing them) to generate high-level intelligence about the overall system.
- **Application domain:** Represents the collection of functions responsible for implementing application logic (including logic, rules, APIs and UIs) and, ultimately, the target business functionalities.
- **Business domain:** Represents the collection of functions responsible for supporting business processes and business-related procedures (e.g., Manufacturing Execution Systems, billing and payment, work planning and scheduling systems).

The above are visualised in Figure 52, along with identified crosscutting functions (i.e., enabling functions needed across functional components) and system characteristics (i.e., how well the system works).

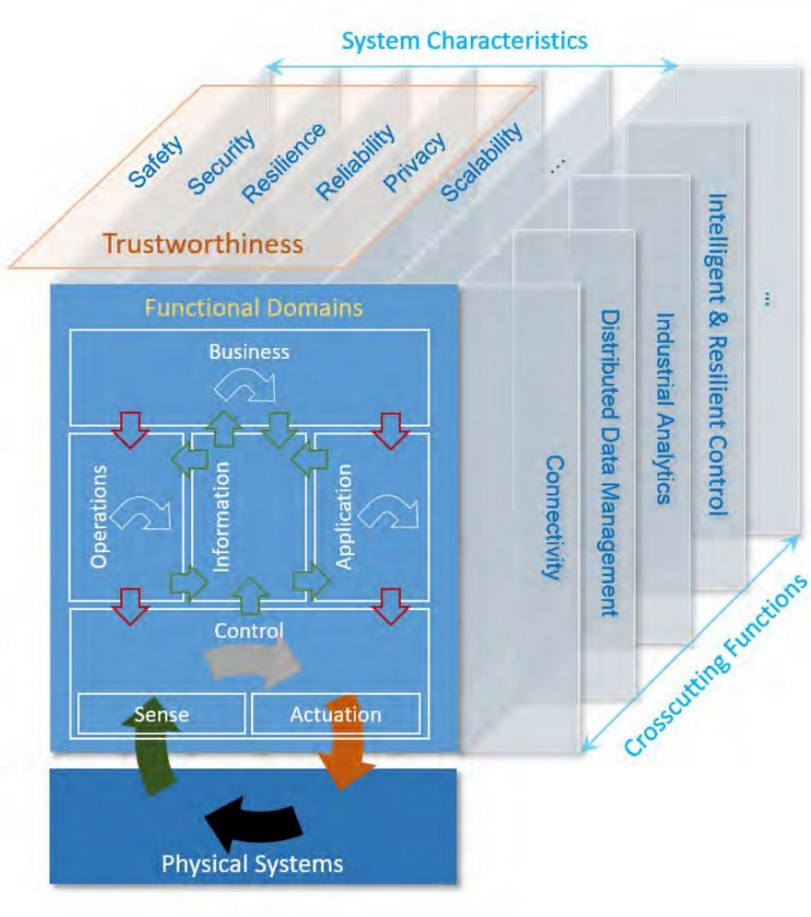

Figure 52. The Industrial Internet Reference Architecture's Functional Viewpoint and its relationship with crosscutting functions and system characteristics.

Considering the above, a mapping between the complex view provided in Figure 52 and the IntellIoT high level architecture is provided in Figure 53. In more detail:

- The Human-in-the-loop enablers which provide user interfacing capabilities and integrate application and part of the business logic are mapped to the Business and Application functional domains.
- The Trustworthiness enablers are directly mapped to the Trustworthiness system characteristic also identified as important within the IIRA.
- The Collaborative IoT pillar is mapped to the Operations and Information functional domains, as it provides part of the provisioning and optimisation capabilities, along with aggregation and intelligence features of IntellIoT.
- The Dynamic Infrastructure Management is mapped to Operations (as it provides part of the provisioning, management, monitoring and optimization capabilities) and also to the intelligent & resilient control crosscutting function, as it provides such features from a compute infrastructure perspective.

- IntellIoT's Networking Infrastructure is directly mapped to the Connectivity crosscutting function of the IIRA, but also to the intelligent & resilient control, as it also provides such features from a network perspective.
- IntellIoT's UC compute infrastructure is mapped at its higher layer to both the Control and Operations (considering the control and operations features provided by gateways and other resource-rich edge devices), and the lower edge device layer is mapped to the control functional domain of the IIRA, as it comprises mostly sensing and actuating capabilities.

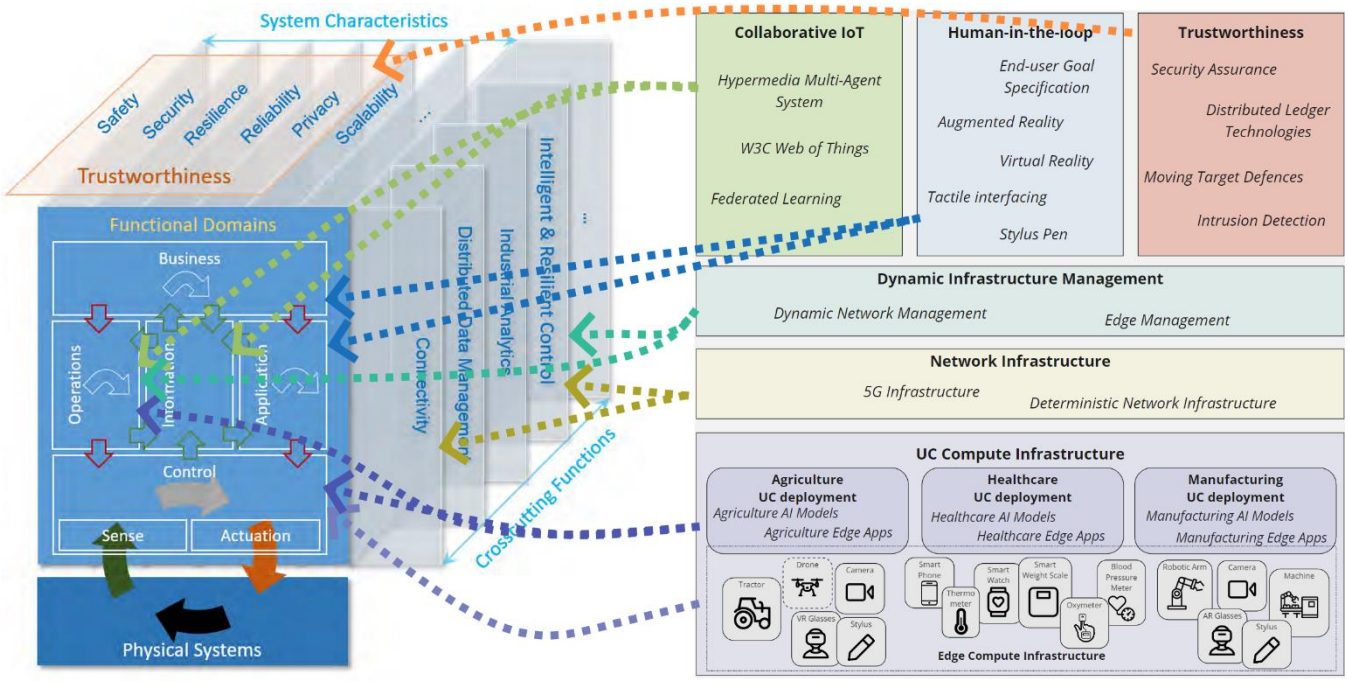

Figure 53. IntellIoT's high level architecture (right) mapped to IIRA's Functional Domains, Crosscutting Functions and System Characteristics (left)

5.4 IDC/European Commission – Edge Taxonomy

Besides the architectural approaches mentioned above, efforts to differentiate between and clarify the different concepts and approaches regarding Edge computing are also interesting. One example of such a classification is the edge taxonomy in Figure 54.

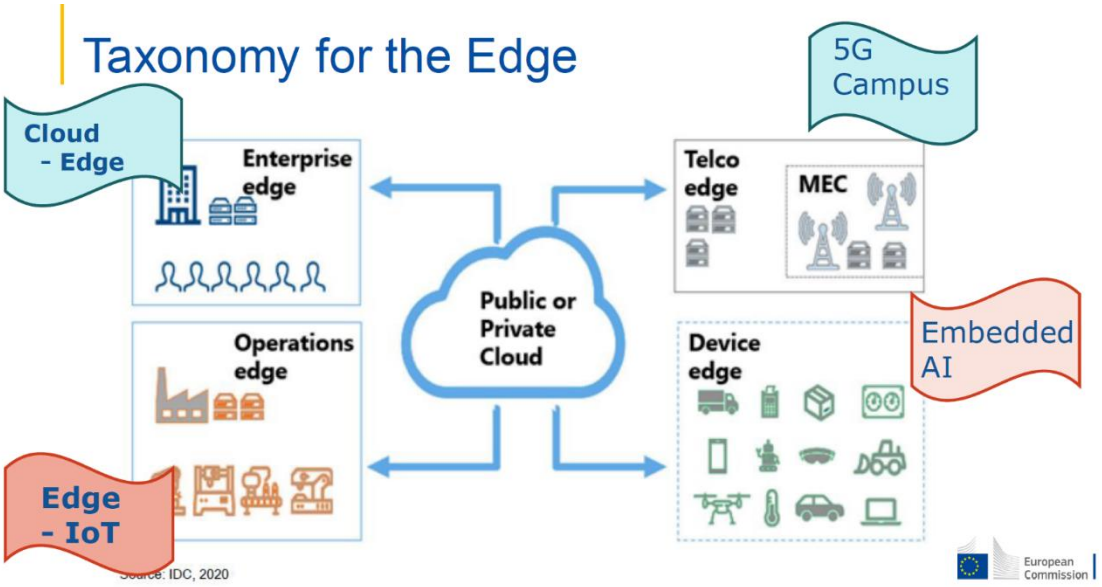

Figure 54. Taxonomy for the Edge (Source: IDC & European Commission)

This taxonomy, originally provided by the IDC and elaborated upon by the EC, differentiates between the following types of Edge:

a) **Enterprise Edge / Cloud - Edge:** In this category, the local edge infrastructure is the priority to execute software programs, however, a private or public cloud (i.e., a remote data centre) is involved in case the program requires too many resources that cannot be found on the edge, or the program needs to be accessible from multiple local environments. This, setup can be meaningful in enterprise environments, e.g., where a remote office or branch office (ROBO) is equipped with local infrastructure (e.g., from simple printing services to simple distributed business applications) and has little to no IT staff to take care of them. More complex applications are then off-loaded to the cloud. Nevertheless, also, for example, in manufacturing plants such a combination of edge and cloud can be meaningfully employed.

b) **Operations Edge / Edge - IoT:** In this category, the local edge infrastructure is integrated with more constrained IoT devices (e.g., sensors and actuators on and around machinery). This converges IT and Operational Technology (OT) and therefore enables for example AI-driven plant automation applications with low latency.

c) **Telco Edge / 5G Campus:** In this category, a local area or "campus" (e.g., the area of an industrial complex, a stadium, a hospital or university) is equipped with private or public 5G network coverage. The 5G network incorporates an edge infrastructure deployment, e.g., through Multi-access Edge Computing (MEC). The MEC supports the execution of edge apps by the clients within the 5G campus.

d) **Device Edge / Embedded AI:** In this category, intelligent end devices are featuring, for example, local analytics and embedded business logic directly through their own computational capacities. In this case, no further edge or cloud infrastructure for offloading these tasks is required. The devices embed the execution of AI.

A high-level mapping of the above to the IntellIoT architecture are provided in Figure 55. For the most part, Human-in-the-loop and Collaborative IoT enablers can be mapped to both Enterprise and Operations Edge (the former mostly to

Enterprise Edge and the latter mostly to Operations Edge, as they operate on different levels of abstraction). The Trustworthiness part can be considered vertical to these concepts (I.e., can be applied and, thus, mapped to all logical entities), so its mapping is omitted for the sake of brevity and to avoid having a tangled figure. The Dynamic Infrastructure management capabilities can be mostly mapped to the Operations Edge, while a straightforward mapping from Network Infrastructure to Telco Edge can also be made. Finally, the higher layer of the UC compute infrastructure can be mapped to the Operations Edge, while the Edge end devices themselves can be linked to the Device Edge in an unambiguous manner.

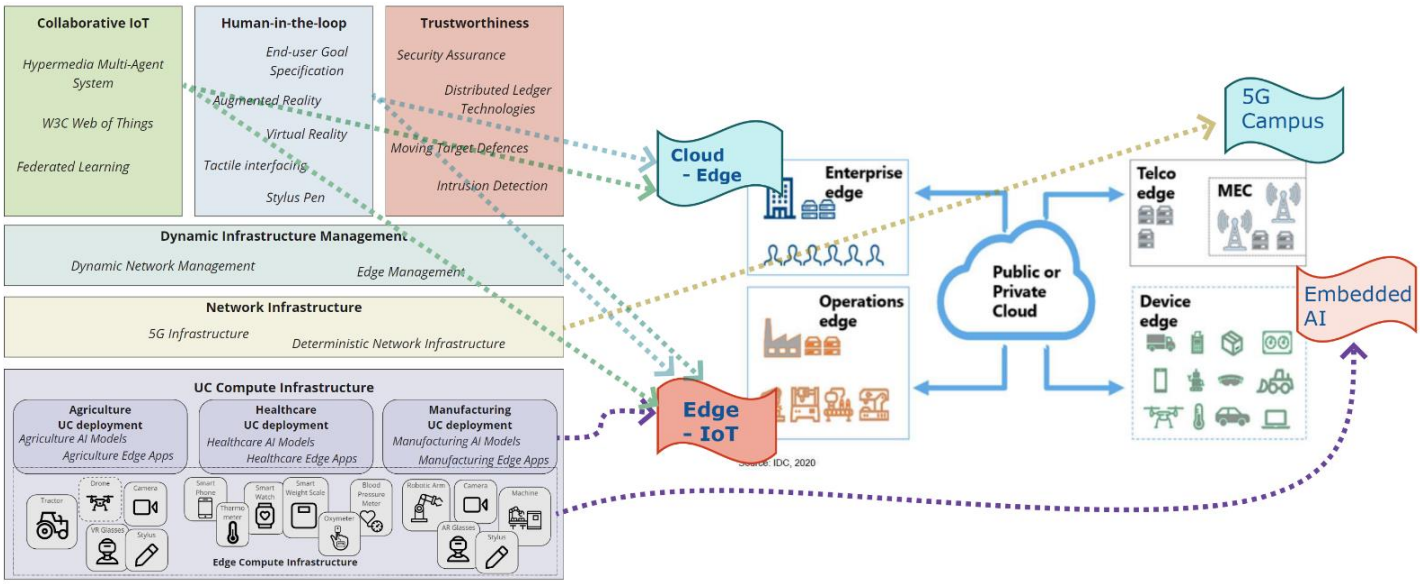

Figure 55. IntellIoT's high level architecture (left) mapped to the Edge Taxonomy (right).

Also interesting is the mapping of the project's UCs to these Edge types. As can be seen in Figure 56, IntellIoT's heterogeneous use cases each offer a different mapping, while, as a whole, the project's use cases provide a full coverage of the Edge types.

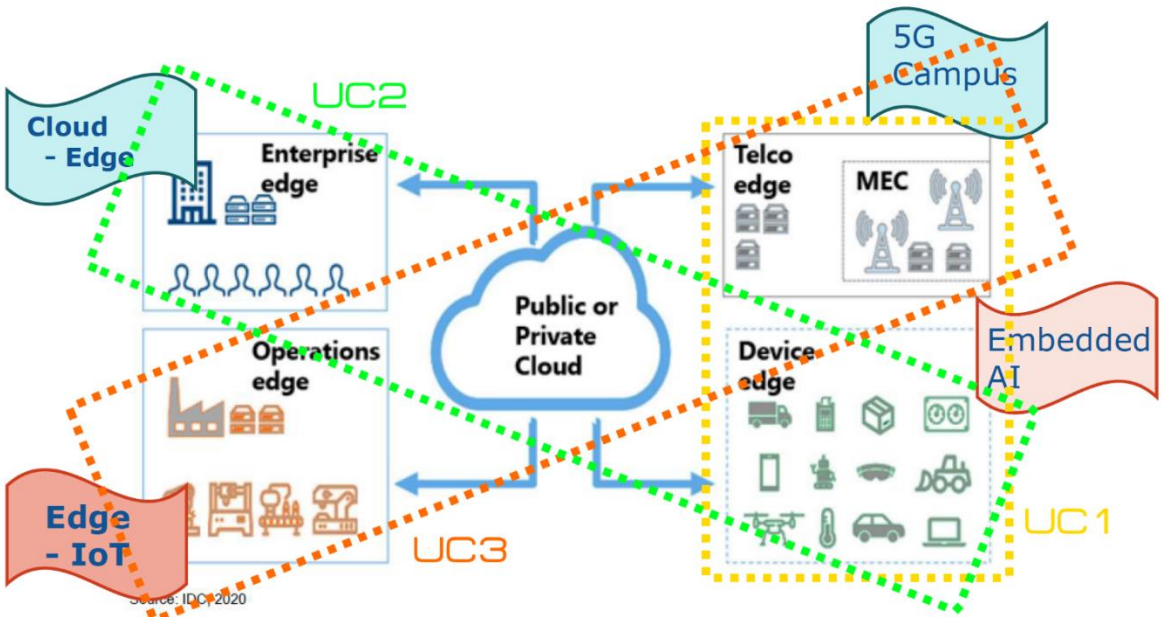

Figure 56. Mapping of IntelllioT's Use Cases to the Edge Taxonomy

More specifically, UC1 (Agriculture) with its focus on the edge devices (tractor, drone) and the support of the 5G infrastructure, can be mainly mapped to Telco and Device Edge. Meanwhile, UC2 (Healthcare) with its emphasis on hospital backend infrastructures (e.g., patient data repository) and patient Edge devices (e.g., wearables), can be mostly mapped to Enterprise and Device Edge. Finally, UC3 (Manufacturing), being dominated by the coexistence and interplay of OT and IT devices, in a 5G-enabled industrial environment, can be mapped to Operations and Telco edge, mainly.

6 CONCLUSIONS AND NEXT STEPS

Being the final output of the architecture task of the project (i.e., Task 2.3 - "Architecture specification & interoperability"), this deliverable provided a detailed specification of the final version of IntellIoT's architecture, providing the foundations for the development, integration, and delivery of the final version of the IntellIoT framework.

Given the foundational role of this deliverable, the aim was to provide a holistic coverage of architectural aspects and related topics that the project will consider. Therefore, a multi-faceted architecture specification methodology was selected by the consortium following the "4+1" architectural view model, featuring Logical, Process, Development and Physical views, along with a use cases view. This not only allowed defining different views of the architecture tailored to different stakeholders (e.g., end users, software developers, system integrators, system engineers), but also allowed to differentiate between core, use case agnostic parts of the architecture, and use case -specific additions, variations and design choices that were made to support the project's three use cases. Regarding the architecture itself, a key decision by the consortium was to design it in a way that prominently features the three pillars of the project (i.e., Collaborative IoT, Human-in-the-loop and Trustworthiness), aligning it to the core elements of the IntellIoT concept.

In addition to the abovementioned architectural views and the associated detailed specifications of the IntellIoT components (key functionalities, underlying technologies, interfacing etc.), the deliverable also documents the derived technical requirements that said components must satisfy. Furthermore, it provides a mapping of involved components and associated technologies to the different requirements, as well as to the overarching project objectives. This aims to showcase the potential full coverage of said elements by the consortium's design decisions. Finally, a study of prominent reference IoT architecture initiatives and their mapping to IntellIoT is included, to validate the alignment of the project's approach with the current state of the art in the IoT architectural landscape.

All of the above have been updated in this final version of the deliverable, building upon the material presented in D2.3, but integrating feedback from developing the components (within WP3 & WP4), the testing, demonstration, validation, and evaluation activities of Cycle 1 (within WP5), and the updated Cycle 2 material defined within D2.4 & D2.5. Furthermore, this final deliverable features alignment with refinements and consistency updates to the various diagrams comprising the architecture to bring them closer to the actual implementation, updates to the requirements, and improved coverage and alignment with the project's objectives and requirements, as informed by the Cycle 1 evaluation & the previously-mentioned early Cycle 2 outputs.

In terms of next steps, this deliverable will guide further efforts within Cycle 2 of the project, including the development of the final version of the components comprising IntellIoT within WP3 and WP4, as well as the final integration, demonstration, and validation efforts, to be carried out within WP5.

REFERENCES

[1] Kruchten, P. B. "The 4+1 View Model of Architecture', IEEE Software, vol.12, no.6, pp.42-50, Nov. 1995. (doi:10.1109/52.469759)

[2] European Comission, Independent High-Level Expert Group on Artificial Intelligence, "Ethics Guidelines For Trustworthy AI", Apr. 2019. https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

[3] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017, April). Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics (pp. 1273-1282). PMLR.

[4] World Wide Web Consortium's (W3C), "Web of Things (WoT) Architecture", W3C Recommendation, Apr. 2020. https://www.w3.org/TR/wot-architecture/

[5] Alliance for the Internet of Things Innovation (AIOTI), WG03 Standardisation, "High Level Architecture (HLA)", Release 5.0, Dec. 2020. https://aioti.eu/wp-content/uploads/2020/12/AIOTI_HLA_R5_201221_Published.pdf

[6] Industrial Internet Consortium (IIC), "Industrial Internet Reference Architecture v1.9", Technical Report, Jun. 2019. https://www.iiconsortium.org/pdf/IIRA-v1.9.pdf

[7] International Telecommunication Union, Telecommunication Standardization Sector's (ITU-T), Recommendation Y.4000/Y.2060 (06/2012), "Overview of the Internet of things", Jun. 2012. http://handle.itu.int/11.1002/1000/11559

[8] Big Data Value Association's (BDVA) Strategic Research Innovation Agenda (SRIA), "European Big Data Value Strategic Research and Innovation Agenda", Version 4.0, Oct. 2017. https://bdva.eu/sites/default/files/BDVA_SRIA_v4_Ed1.1.pdf

[9] oneM2M, TS-0001-V4.11.0, "Functional Architecture", Version 4.11.0, Jun. 2021. https://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=33796

[10] IDC, "Edge Computing: Not All Edges are Created Equal", Jun. 2020. https://blogs.idc.com/2020/06/01/edge-computing-not-all-edges-are-created-equal/

[11] ISO/IEC/IEEE, "ISO/IEC/IEEE 42010:2011 Systems and software engineering - Architecture description", 2011. https://www.iso.org/standard/50508.html