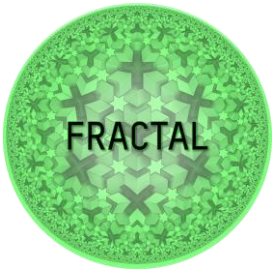

D8.2 - System Requirements

Deliverable Id:	D8.2
Deliverable Name:	System Requirements
Status:	Final
Dissemination Level:	Public
Due date of deliverable:	2022/12/31
Actual submission date:	2022/12/31
Work Package:	WP8 "Case Studies, Specification, Benchmarking & Justification File"
Organization name of lead contractor for this deliverable:	Aitek
Authors:	Stefano Delucchi (Aitek) Jon Mikel Olmos (CAF) Luigi Pomante (UNIVAQ) Markus Postl (VIF) Artur Kaufmann (BEE) Nadia Caterina Zullo Lasala (ROT) Mattia Modugno (ROT) Enrico Ferrari (Rulex)
Reviewers:	Paolo Burgio (UNIMORE) Martin Matschnig (SIEM)

Abstract

This deliverable defines the requirements and the mechanisms used by the different components to exchange information within each use case. UC5, UC6, UC7, UC8 are presented in different chapters highlighting the requirements (coming from WP2 and from further refinement). Moreover, the UC architecture and components as well as the data interfaces are presented and described by means of diagrams. D8.2 represents therefore a set of specifications to properly implement the use cases.

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 877056

Co-funded by the Horizon 2020 Programme of the European Union under grant agreement No 877056.

Contents

1 History

Version	Date	Modification reason	Modified by
V 0.1	14/10/2022	Table of Content	Aitek
V1.0	28/10/2022	1st contribution (UC6)	Aitek
V1.1	10/11/2022	2nd contribution	Aitek, CAF, UNIVAQ, VIF, BEE
V1.2	24/11/2022	3rd contribution	Aitek, CAF, UNIVAQ, VIF, BEE
V2.0	08/12/2022	Ready for internal review	Aitek, CAF, UNIVAQ, VIF, BEE
V3.0	21/12/2022	Final version ready for submission	Aitek, CAF, UNIVAQ, VIF, BEE

2 Summary

The goal of this deliverable is to provide a report on use case requirements and data interfaces to define how use case building blocks (i.e., use case components) communicate in each use case. It is an important document that will be used as a reference during use case implementation (Task 8.2). Moreover, it is tightly related to other FRACTAL Work Packages (WP) as it refines requirements originally defined in WP2 and defines interfaces between use case components that are built on top of FRACTAL components developed in WP3, 4, 5 and 6. In particular, such interfaces have been defined in detail to avoid incompatibilities between components following a common approach. A thorough analysis has been made by use case leaders to prevent any inconsistencies in the description. Giving that, this deliverable can be considered a sort of use case specification to be used to proper configure components and data interfaces during use case implementation.

3 Introduction

This deliverable is part of WP8 titled "Case Studies, Specification, Benchmarking & Justification File". For sake of completeness, this introduction starts with a brief summary of this WP then it continues with the description of the methodologies used to collect and report information and finally it presents the structure and the content of the rest of the deliverable.

The objective of **Work Package 8** is to specify, develop and integrate solutions for four different industrial use cases representative of different application areas. They are:

- VAL-UC5, led by CAF and entitled "Increasing the safety of an autonomous train through AI techniques".
- VAL-UC6, jointly led by AITEK and UNIVAQ and entitled "Intelligent Totem - Elaborate data collected using heterogeneous technologies".
- VAL-UC7, led by VIF and entitled "Autonomous SPIDER Robot for implementing safe movements".
- VAL-UC8, led by BEE and entitled "Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse".

The goal of this WP is to implement use case and prototype in each use case, to demonstrate and validate FRACTAL technological results (i.e., components, methodologies, tools and solutions developed during the project) in relevant applicative domains and real-word situations. To reach this goal, this WP and in particular its first task (T8.1 "Case study Coordination") is focused on ensuring a fruitful coordination between use case providers (WP8), technology providers (WP3-WP6). D8.1 ("Specification of Industrial validation Use Cases") represents the output of T8.1 also defining the methodologies and the metrics to evaluate use case progress and results. This document, Deliverable D8.2 (referred to T8.2 and T8.3, respectively "Case Study Benchmark Specification" and "Case Study Implementation") is quite complementary with respect to the previous one, describing use case requirements and data interfaces, meaning explaining how to integrate use case components and how they cooperate in each use case demonstrator. Finally, D8.3 ("Evaluation Results"), related to task T8.4 ("Case Study justification File") contains an evaluation of the activities done in each use case.

This **deliverable is organized per use case** with a specific chapter dedicated for each use case. As discussed, and agreed in WP8 meetings, we followed a hierarchical approach in which each use case leader is responsible for his chapter. Aitek, as deliverable responsible, provided the Table of Content (V0.1) and initial example by filling UC6 chapter. Use case leaders adapted the structure of their chapters according to their specific needs.

In more detail, chapters are divided in different subsections with different focus:

- Functional and non-functional requirements:

  - Requirements from WP2. This represents the starting point, e.g., the list of requirements defined in WP2 and reported in D2.1 ("Platform Specification").
  - Requirements refinement during Use Cases specification. This section documents the requirements refinement process done in the scope of WP8 for each use case. In more details it contains the list of requirements *i)* confirmed, *ii)* removed, *iii)* modified, *iv)* added and it ends with the list of consolidated requirements.
- Data interfaces:
  - Use case Architecture and Components. UC components description and integration within a use case specific architecture, focusing on the final demonstrators
  - Use case diagram, used to describe the interaction between use case component within the demonstrator architecture
  - Data Interfaces to describe data to be exchanged, their formats and protocols used.

It is worth noticing that defining data interfaces is a significant part of this document. Therefore, we defined a standardized way to describe them as reported here below (Figure 1).

- Data INPUT_#:
  - Source ID:
  - Source Name:
  - Data type:
  - Periodic / event-driven:
  - Protocol definition (physical, logical / messages format):
    - Physical:
    - Logical:
    - Message format:
  - Other relevant info:
- Data OUTPUT_#:
  - Destination ID:
  - Destination Name:
  - Data type:
  - Periodic / event-driven:
  - Protocol definition (physical, logical / messages format):
    - Physical:
    - Logical:
    - Message format:
  - Other relevant info:

Figure 1 – Template to describe data interfaces for each use case component

4 VAL-UC5 Increasing the safety of an autonomous train through AI techniques

4.1 Functional and non-functional requirements

4.1.1 Requirements from WP2

Req ID	Description
REQ_UC5_01	The edge node shall support OpenCV library
REQ_UC5_02	The edge node shall have ONNX Interpreter
REQ_UC5_03	The HW accelerators might be compatible with Tensorflow
REQ_UC5_04	The edge node shall provide multi-core technology with at least 4 CPU cores
REQ_UC5_05	The edge node shall handle multi-threading applications
REQ_UC5_06	The edge node shall have at least 60GFLOPS
REQ_UC5_07	The edge node should provide at least 16GB DDR RAM
REQ_UC5_08	The edge node shall incorporate HW acceleration
REQ_UC5_09	The edge node shall incorporate IO interfaces: 2xGbit Eth, 2xUSB 3.0, 1xHDMI (and their Linux Drivers)
REQ_UC5_10	The edge node shall have Linux OS
REQ_UC5_11	The edge node release shall have C++ compiler/ cross-compile toolchain
REQ_UC5_12	The edge node should be compliant with non-functional railway equipment requirements described in EN50155, EN50125, EN45545, EN50121, UNE EN61373: temperature, humidity, vibration, electromagnetic compatibility, fire and electrical protection
REQ_UC5_13	The edge node shall support real-time performance for UC5 functionality providing processing capability of 100ms/10fps

Table 1 – UC5 requirements defined in WP2

4.1.2 Requirements refinement during Use Cases specification

4.1.2.1 Requirements confirmed

Req ID	Description
REQ_UC5_01	The edge node shall support OpenCV library
REQ_UC5_02	The edge node shall have ONNX Interpreter
REQ_UC5_03	The HW accelerators might be compatible with Tensorflow
REQ_UC5_04	The edge node shall provide multi-core technology with at least 4 CPU cores
REQ_UC5_05	The edge node shall handle multi-threading applications
REQ_UC5_06	The edge node shall have at least 60GFLOPS

REQ_UC5_07	The edge node should provide at least 16GB DDR RAM
REQ_UC5_08	The edge node shall incorporate HW acceleration
REQ_UC5_09	The edge node shall incorporate IO interfaces: 2xGbit Eth, 2xUSB 3.0, 1xHDMI (and their Linux Drivers)
REQ_UC5_10	The edge node shall have Linux OS
REQ_UC5_11	The edge node release shall have C++ compiler/ cross-compile toolchain
REQ_UC5_12	The edge node should be compliant with non-functional railway equipment requirements described in EN50155, EN50125, EN45545, EN50121, UNE EN61373: temperature, humidity, vibration, electromagnetic compatibility, fire and electrical protection
REQ_UC5_13	The edge node shall support real-time performance for UC5 functionality providing processing capability of 100ms/10fps

Table 2 – UC5 requirements confirmed

4.1.2.2 Requirements removed

No requirement removal was defined.

4.1.2.3 Requirements modified

No requirement modification was performed, REQ_UC5_12 can be set to optional taking into account that industrial qualification is out of project scope.

4.1.2.4 Requirements added

No new requirements are defined in WP8 for UC5

4.1.2.5 Consolidated list of requirements

Req ID	Description
REQ_UC5_01	The edge node shall support OpenCV library
REQ_UC5_02	The edge node shall have ONNX Interpreter
REQ_UC5_03	The HW accelerators might be compatible with Tensorflow
REQ_UC5_04	The edge node shall provide multi-core technology with at least 4 CPU cores
REQ_UC5_05	The edge node shall handle multi-threading applications
REQ_UC5_06	The edge node shall have at least 60GFLOPS
REQ_UC5_07	The edge node should provide at least 16GB DDR RAM
REQ_UC5_08	The edge node shall incorporate HW acceleration
REQ_UC5_09	The edge node shall incorporate IO interfaces: 2xGbit Eth, 2xUSB 3.0, 1xHDMI (and their Linux Drivers)
REQ_UC5_10	The edge node shall have Linux OS
REQ_UC5_11	The edge node release shall have C++ compiler/ cross-compile toolchain

REQ_UC5_12	The edge node should be compliant with non-functional railway equipment requirements described in EN50155, EN50125, EN45545, EN50121, UNE EN61373: temperature, humidity, vibration, electromagnetic compatibility, fire and electrical protection
REQ_UC5_13	The edge node shall support real-time performance for UC5 functionality providing processing capability of 100ms/10fps

Table 3 – UC5 consolidated list of requirements

4.2 Data interfaces

Use Case 5 presents two main functionalities and one additional feature:

- Automatic Accurate Stop: Landmark detection marking the right stopping location and notifying to ATO (Automatic Train Operation) system about the remaining distance until the stop.
- Safe Passenger Transfer: Passenger detection on the train platform surroundings and notify ATO systems when the zone is clear and train can close doors and depart.
- Automatic Software Update: At train startup, software version is pulled from the cloud repository hosting the latest release for above functionalities SW.

4.2.1 Use case Architecture and Components

In Use Case 5 the proposed architecture is based on a high performance edge node which executes all real time operations required for the functionalities in scope. Real time operations must be executed on the edge in order to reduce the latency that the cloud connection introduces.

A cloud node is introduced for hosting software releases that are ready to be installed on the edge node by the Automatic Update Service.

The component input/output definition is divided into two configurations, the test setup and the production setup. In the test setup the input is simulated from a test database which allows to perform testing without train integration and the Non-Functional requirements compliance that implies.

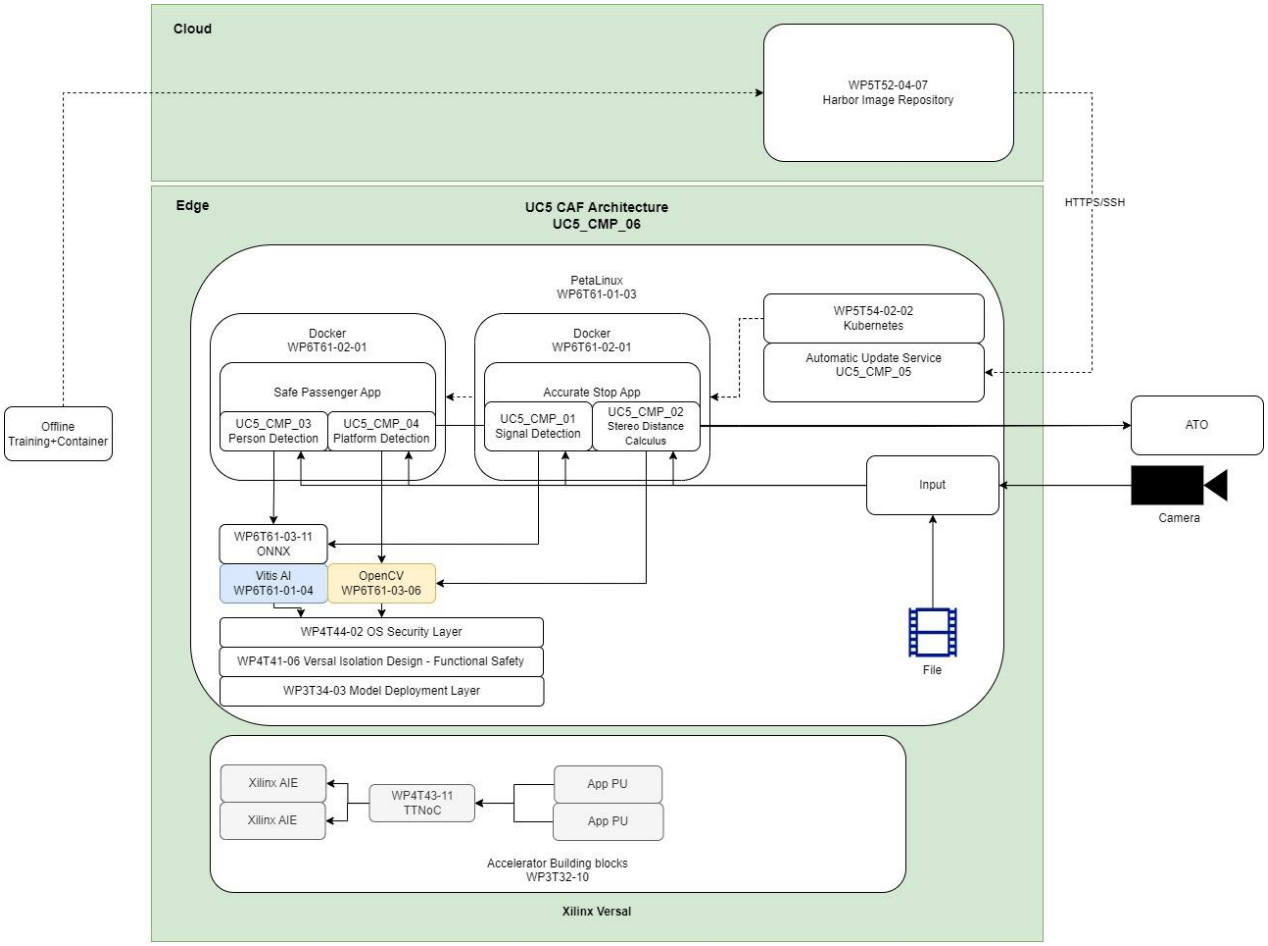

Figure 2 – Architecture of Use Case 5

UC5_CMP_01: Signal Detection

The Signal Detection component is based on a YOLOv3 detector trained with custom signal dataset. It performs detection using frontal camera images as input to detect the landmark that marks the exact stopping location for the train.

- Data INPUT:
  - Source Name: Test Database/Front Camera 1
  - Data type: Video RGB 1280x960
  - Periodic / event-driven: Periodic 10 fps
  - Protocol definition (physical, logical / messages format):
    - Physical: USB 3.0
    - Logical: GenTL API
    - Message format: Frame Raw RGB8
  - Other relevant info: USB3.0 Camera with GenTL standard transport layer. Only for production deployment setup
- Data OUTPUT:
  - Destination ID: ATO
  - Destination Name: ATO
  - Data type: UC5_IF_09 Custom UDP data

- Periodic / event-driven: Periodic (one per analysed frame)
- Protocol definition:
  - Physical: Eth
  - Logical: UDP
  - Message format: Custom UC5_IF_09
- Other relevant info: detection class and 2D location

UC5_CMP_02: Stereo Distance Calculus

Distance estimation based on SGBM OpenCV algorithm that takes front1 and front2 calibrated cameras as input and generates a depthmap with reduced resolution. Depthmap can be used by UC5_CMP_01 for estimating distance to detected landmark.

- Data INPUT1:
  - Source Name: Test Database/Front Camera 1
  - Data type: Video RGB 1280x960
  - Periodic / event-driven: Periodic 10 fps
  - Protocol definition (physical, logical / messages format):
    - Physical: USB 3.0
    - Logical: GenTL API
    - Message format: Frame Raw RGB8
  - Other relevant info: USB3.0 Camera with GenTL standard transport layer. Only for production deployment setup
- Data INPUT2:
  - Source Name: Test Database/Front Camera 2
  - Data type: Video RGB 1280x960
  - Periodic / event-driven: Periodic 10 fps
  - Protocol definition (physical, logical / messages format):
    - Physical: USB 3.0
    - Logical: GenTL API
    - Message format: Frame Raw RGB8
  - Other relevant info: USB3.0 Camera with GenTL standard transport layer. Only for production deployment setup
- Data OUTPUT:
  - Destination ID: ATO
  - Destination Name: ATO
  - Data type: UC5_IF_09 Custom UDP data
  - Periodic / event-driven: Periodic (one per analysed frame)
  - Protocol definition:
    - Physical: Eth
    - Logical: UDP
    - Message format: Custom UC5_IF_09
  - Other relevant info: distance to detection

UC5_CMP_03: Passenger/Person Detection

Passenger detection based on yolov3 standard trained. Takes rear mirror cameras as input and generates detections corresponding to passengers that can , aftwerwards, be used to define train surroundings clearance.

- Data INPUT1:
  - Source Name: Test Database/Rear Camera 1
  - Data type: Video RGB 1920x1080 Stream
  - Periodic / event-driven: Periodic 10 fps
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: RTSP
    - Message format: Encoded H264 Stream
  - Other relevant info: RTSP Surveillance camera
- Data INPUT2:
  - Source Name: Test Database/Rear Camera 2
  - Data type: Video RGB 1920x1080 Stream
  - Periodic / event-driven: Periodic 10 fps
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: RTSP
    - Message format: Encoded H264 Stream
  - Other relevant info: RTSP Surveillance camera
- Data OUTPUT:
  - Destination ID: ATO
  - Destination Name: ATO
  - Data type: UC5_IF_10 Custom UDP data
  - Periodic / event-driven: Periodic (one per analysed frame)
  - Protocol definition:
    - Physical: Eth
    - Logical: UDP
    - Message format: Custom UC5_IF_10
  - Other relevant info: Custom packet defining environment status, number of passengers on danger area and number of passengers near danger area

UC5_CMP_04: Platform Detection

Platform area definition for detecting proximity level of passengers to train. Uses preconfigured position rules on image.

- Data INPUT1:
  - Source Name: Test Database/Rear Camera 1
  - Data type: Video RGB 1920x1080 Stream
  - Periodic / event-driven: Periodic 10 fps
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth

        - Logical: RTSP
        - Message format: Encoded H264 Stream
    - Other relevant info: RTSP Surveillance camera
- Data INPUT2:
    - Source Name: Test Database/Rear Camera 2
    - Data type: Video RGB 1920x1080 Stream
    - Periodic / event-driven: Periodic 10 fps
    - Protocol definition (physical, logical / messages format):
        - Physical: Eth
        - Logical: RTSP
        - Message format: Encoded H264 Stream
    - Other relevant info: RTSP Surveillance camera
- Data OUTPUT:
    - Destination ID: ATO
    - Destination Name: ATO
    - Data type: UC5_IF_10 Custom UDP data
    - Periodic / event-driven: Periodic (one per analysed frame)
    - Protocol definition:
        - Physical: Eth
        - Logical: UDP
        - Message format: Custom UC5_IF_10
    - Other relevant info: Custom packet defining environment status, number of passengers on danger area and number of passengers near danger area

UC5_CMP_05: Automatic update service

Linux service configured at OS startup. Pulls container from Cloud image repository containing main UC applications. Uses ssh/https connection with cloud node to pull SW through secure channel.

- Data INPUT:
    - Source Name: Harbor image repository
    - Data type: Docker Container
    - Periodic / event-driven: event-driven (system startup)
    - Protocol definition (physical, logical / messages format):
        - Physical: 4G
        - Logical: ssh/https
        - Message format: Docker pull
    - Other relevant info:
- Data OUTPUT:
    - Destination ID: Edge node Linux OS
    - Destination Name: Edge node Linux OS
    - Data type: Docker container
    - Periodic / event-driven: event-driven (system startup)
    - Protocol definition :
        - Physical: Internal memory

- Logical:
- Message format:

- Other relevant info: Writes the pulled docker containers on fixed paths and launches them

4.2.2 Use case diagram

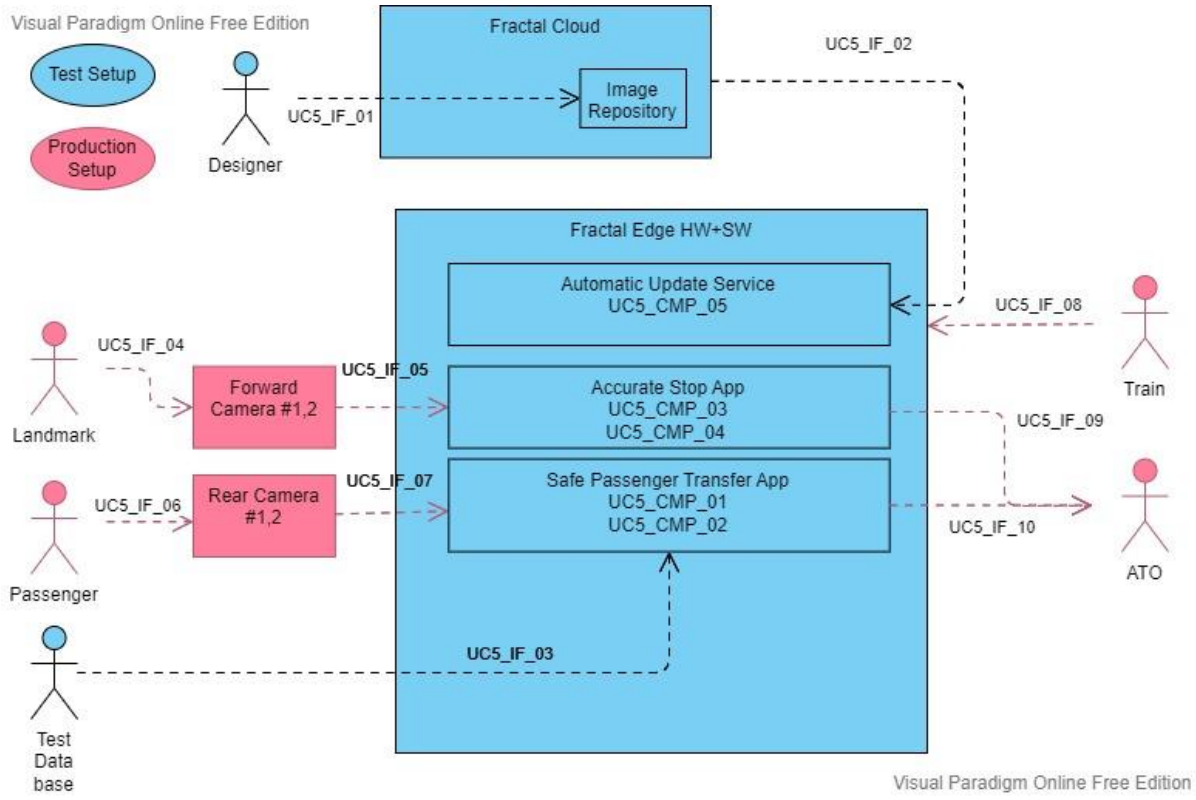

Figure 3 – Interface Diagramof UC5

4.2.3 Data interfaces

4.2.3.1 UC5_IF_01: Designer – Image Repository

The Harbor Image Repository is expected to store software releases inside a docker container. Image containers for both applications (safe passenger transfer and accurate stop) are uploaded to the Harbor Image Repository located in the fractal cloud by a team designer that generates the SW release.

4.2.3.2 UC5_IF_02: Image Repository– Automatic Update Service (UC5_CMP_05)

This interface is based on Docker native pull system over ssh/https connection to guarantee security requirements.

4.2.3.3 UC5_IF_03: Test Database– Accurate Stop App/Safe Passenger Transfer App

This interface acts as a simulator for real camera interfaces allowing to test the functionalities without integration on train. It provides videos recorded in real environment for front cameras and rear mirror cameras.

4.2.3.4 UC5_IF_04: Landmark– Front Camera #1,2

A Landmark is recorded on calibrated front cameras #1,2 with certain camera parameters affecting brightness, exposure and resolution (1280x960 RGB)

4.2.3.5 UC5_IF_05: Front Camera #1,2 – Accurate Stop App

A camera-grabbed image is delivered using USB3.0 GenTL interface sending raw 1280x960 RGB 8bit frames at 10 fps.

4.2.3.6 UC5_IF_06: Passenger– Rear Camera #1,2

Passengers within rear camera range are grabbed with not-configurable camera parameters set at train factory delivery.

4.2.3.7 UC5_IF_07: Rear Camera #1,2 – Safe Passenger Transfer App

Rear mirror camera images are served through RTSP server on train comfort network (private within train context) to the cabin display for the driver and to the Safe Passenger Transfer Application. The streaming is configured on demand and uses a H264 encoded stream.

4.2.3.8 UC5_IF_08: Train – Fractal Edge Node

The train starts the fractal edge node with direct power supply. TCMS (Train Control& Management System) system activates the power supply for the fractal edge node during train start up sequence.

4.2.3.9 UC5_IF_09: Accurate Stop App – ATO

This custom interface transfers plain data over UDP with a header. It is sent asynchronously every time Accurate Stop Application processes a frame. The structure describing the header can be seen in Table 4 and the structure describing the payload is shown in Table 5.

HEADER UC5_IF_09		
Field	**Description**	**Length**
NID_PACKET	Packet type identifier	Unsigned 16 bits
LENGTH	Packet length including header	Unsigned 32 bits
TIMESTAMP	Packet timestamp (not utc, sys timestamp incremental)	Unsigned 32 bits

Table 4 – Header for UC5_IF_09

LANDMARK_PACKET(NID_PACKET=1) UC5_IF_09			
Field		**Description**	**Length**
N_ITER		Iterator counter (max 8 elements)	Unsigned 8 bits
	LANDMARK_TYPE	0: Station lines platform start 1: Station lines platform end 2-255:Spare	Enum 8 bits
	REMAINING_DISTANCE	Remaining distance to landmark in cm	Unsigned 32 bits

Table 5 – Payload packet LANDMARK_PACKET for UC5_IF_09

4.2.3.10 UC5_IF_09: Safe Passenger Transfer App – ATO

This custom interface transfers plain data over UDP with a header. It is sent asynchronously every time the Safe Passenger Transfer Application processes a frame. The structure describing the header can be seen on Table 6 and the structure describing the payload is shown in Table 7.

HEADER UC5_IF_10		
Field	**Description**	**Length**
NID_PACKET	Packet type identifier	Unsigned 16 bits
LENGTH	Packet length including header	Unsigned 32 bits
TIMESTAMP	Packet timestamp (not utc, sys timestamp incremental)	Unsigned 32 bits

Table 6 - Header for UC5_IF_10

PASSENGER_DETECTION_PACKET(NID_PACKET=1) UC5_IF_10		
	Description	**Length**
ENVIRONMENT_STATUS	0: Clear (no passengers in and near danger zone) 1: Potentially dangerous (passengers out of danger zone but close) 2: Dangerous (passengers in danger zone) 3-255: Spare Environment Status regarding passenger presence	Enum 8 bits

Table 7 - Payload packet PASSENGER_DETECTION_PACKET for UC5_IF_10

5 VAL-UC6 Elaborate data collected using heterogeneous technologies (intelligent totem)

5.1 Functional and non-functional requirements

5.1.1 Requirements from WP2

Req ID	Description
REQ_UC6_01	The edge node shall be able to execute TensorFlow-Keras framework models and/or ONNX (https://onnx.ai/).
REQ_UC6_02	The edge node shall OpenCV Library.
REQ_UC6_03	The edge node shall detect user face.
REQ_UC6_04	The edge node shall detect user age.
REQ_UC6_05	The edge node shall detect user gender.
REQ_UC6_06	The edge node shall detect people at totem proximity.
REQ_UC6_07	The edge node shall count people, or track people density, in totem proximity.
REQ_UC6_08	The edge node should detect crowd intensity and variation.
REQ_UC6_09	The edge node shall detect speaker Language.
REQ_UC6_11	The edge node shall select content/info to be provided.
REQ_UC6_11	The edge node shall select the output channel among those available (e.g., video, audio, etc.).
REQ_UC6_12	The edge node shall support TCP/IP protocol; the ideal network protocol to transport messages among the devices shall be MQTT or any other publish/subscribe communication protocol.
REQ_UC6_13	The edge node shall expose a set of APIs which shall allow HTTPs REST calls to and from other nodes, central application and user devices.
REQ_UC6_14	The edge node shall acquire images from at least one HD camera.
REQ_UC6_15	The edge node shall acquire audio signal from at least one microphone.
REQ_UC6_16	The edge node shall support programmable accelerator engines, such as for instance FPGA/programmable logics, or AI engines (e.g., VERSAL's) or in case, GPGPUs.
REQ_UC6_17	The edge node shall have Linux OS such as Ubuntu or Petalinux.
REQ_UC6_18	The edge node shall have a C++ compiler and related standard libraries.
REQ_UC6_19	The edge node shall support wired connectivity (e.g., Ethernet) in order to ensure network stability, it shall have at least 1Gbit Ethernet connection.
REQ_UC6_20	The edge node shall support wireless connectivity (e.g., Wi-Fi).
REQ_UC6_21	The edge node shall have a hardware computing node that allows accelerating convolutional neural networks applications.
REQ_UC6_22	The edge node should control an interactive touchscreen display.
REQ_UC6_23	The edge node should control an audio speaker.

REQ_UC6_24	The edge node shall have a monitoring system able to measure response time of tasks, both implemented on microprocessors and accelerators.
REQ_UC6_25	The edge node shall be able to share the computational workload with other nodes when necessary.
REQ_UC6_26	The edge node shall guarantee that the user's thoughts to remain uninterrupted is not more than 1.0 second
REQ_UC6_27	The edge node should have a modular and scalable architecture to allow an easy and quick integration of new data sources without changing the architecture.
REQ_UC6_28	The edge node shall store data locally in a secure manner.

Table 8 – UC6 requirements defined in WP2

5.1.2 Requirements refinement during Use Cases specification

5.1.2.1 Requirements confirmed

All requirements, except REQ_UC6_07, REQ_UC6_08, and REQ_UC6_11 have been confirmed.

5.1.2.2 Requirements removed

Requirement REQ_UC6_07 has been removed because it is covered by REQ_UC6_06 REQ_UC6_08.

REQ_UC6_07	The edge node shall count people, or track people density, in totem proximity.

Requirement REQ_UC6_11 has been removed because it is no more relevant for the use case (i.e., content is always provided by both output channels and therefore the selection process is not needed).

REQ_UC6_11	The edge node shall select the output channel among those available (e.g., video, audio, etc.).

5.1.2.3 Requirements modified

(List of requirements defined in WP2 and modified in WP8)

REQ_UC6_08 has been modified as reported below to enforce its relevance for our use case.

REQ_UC6_08	The edge node shall estimate crowd intensity.

5.1.2.4 Requirements added

No requirements have been added

5.1.2.5 Consolidated list of requirements

(List of requirements - final)

Req ID	Description
REQ_UC6_01	The edge node shall be able to execute TensorFlow-Keras framework models and/or ONNX (https://onnx.ai/).
REQ_UC6_02	The edge node shall OpenCV Library.
REQ_UC6_03	The edge node shall detect user face.
REQ_UC6_04	The edge node shall detect user age.
REQ_UC6_05	The edge node shall detect user gender.
REQ_UC6_06	The edge node shall detect people at totem proximity.
REQ_UC6_08	The edge node shall estimate crowd intensity.
REQ_UC6_08	The edge node shall estimate crowd intensity.
REQ_UC6_09	The edge node shall detect speaker Language.
REQ_UC6_11	The edge node shall select content/info to be provided.
REQ_UC6_12	The edge node shall support TCP/IP protocol; the ideal network protocol to transport messages among the devices shall be MQTT or any other publish/subscribe communication protocol.
REQ_UC6_13	The edge node shall expose a set of APIs which shall allow HTTPs REST calls to and from other nodes, central application and user devices.
REQ_UC6_14	The edge node shall acquire images from at least one HD camera.
REQ_UC6_15	The edge node shall acquire audio signal from at least one microphone.
REQ_UC6_16	The edge node shall support programmable accelerator engines, such as for instance FPGA/programmable logics, or AI engines (e.g., VERSAL's) or in case, GPGPUs.
REQ_UC6_17	The edge node shall have Linux OS such as Ubuntu or Petalinux.
REQ_UC6_18	The edge node shall have a C++ compiler and related standard libraries.
REQ_UC6_19	The edge node shall support wired connectivity (e.g., Ethernet) in order to ensure network stability, it shall have at least 1Gbit Ethernet connection.
REQ_UC6_20	The edge node shall support wireless connectivity (e.g., Wi-Fi).
REQ_UC6_21	The edge node shall have a hardware computing node that allows accelerating convolutional neural networks applications.
REQ_UC6_22	The edge node should control an interactive touchscreen display.
REQ_UC6_23	The edge node should control an audio speaker.
REQ_UC6_24	The edge node shall have a monitoring system able to measure response time of tasks, both implemented on microprocessors and accelerators.
REQ_UC6_25	The edge node shall be able to share the computational workload with other nodes when necessary.
REQ_UC6_26	The edge node shall guarantee that the user's thoughts to remain uninterrupted is not more than 1.0 second
REQ_UC6_27	The edge node should have a modular and scalable architecture to allow an easy and quick integration of new data sources without changing the architecture.
REQ_UC6_28	The edge node shall store data locally in a secure manner.

5.2 Data interfaces

(This sub-section describes data exchanged among UC components. Focus on data formats, protocols, etc)

5.2.1 Use case Architecture and Components

In this use case we consider two FRACTAL nodes cooperating together, the roof node (on the left side of Figure 4) and the totem node (on the right side of Figure 4).

There is also a broker MQTT to manage the communication between the nodes. This can be executed on an external board with very limited computation capability (e.g., a Raspberry PI) used only for demonstration purposes and out of FRACTAL scope.

The **roof node** consists of a CCTV camera and a FRACTAL node that executes some of the video content analysis tasks, hosting some use case 6 components.

The **totem node** consists of an interactive totem with camera, microphone, speakers touch display and a FRACTAL node that hosts other use case components and executes video content analysis tasks, audio processing and rule-based recommendation to select the multimedia content to be displayed on the totem. In such a node, there is also a Runtime Manager, a component that is in charge of managing all the others. It includes also a load balancer for components offloading, that allocates the AI tasks between the two nodes. In particular, this component acts when a node is not able to perform all the required computation within a predefined time, it checks if there are other nodes capable of taking over the processing.

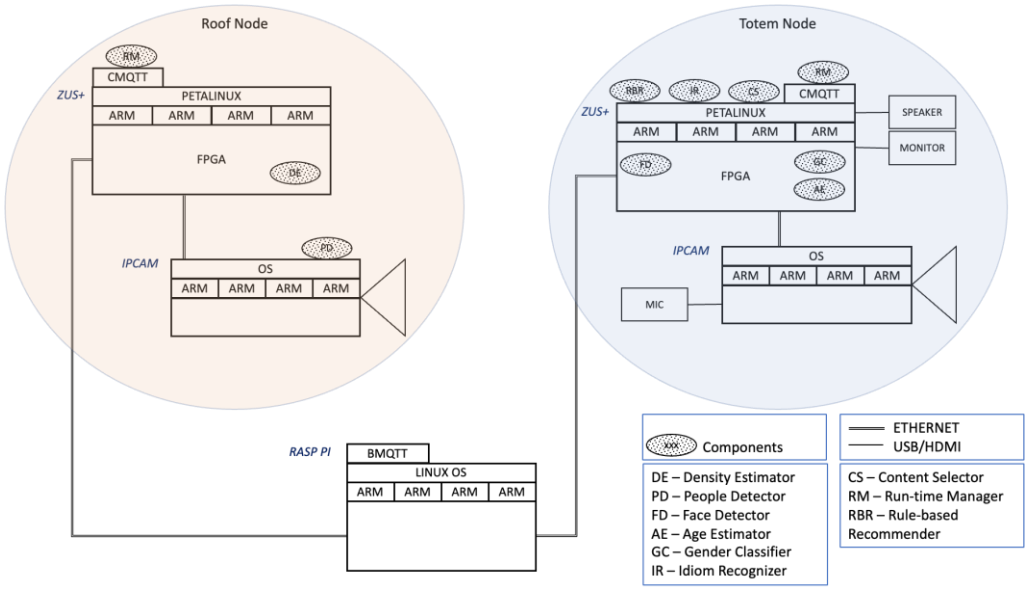

Figure 4 – Architecture of Use Case 6

Here is reported a list of use case components hosted on the two nodes.

- Roof node:
  - Density Estimator, provided by UNIMORE

  - People Detector, provided by AITEK
  - Runtime Manager, provided by ROT
- Totem node:
  - Face Detector, provided by UNIMORE
  - Age Estimator, provided by UNIVAQ
  - Gender Classifier, provided by UNIVAQ
  - Idiom Recognizer, provided by UNIGE
  - Runtime Manager, provided by ROT
  - Rule-based Recommender, provided by RULEX

For what concerns the communication technologies we decided to use wired connection to build an integrated prototype and for demonstration purposes (as we are targeting a lower TRL with respect to a real implementation). Nevertheless, we also considered the possibility of using a wireless network for handling roof-totem communication in a real application aiming TRL 9 after project conclusion. To reach this objective we analysed different available communication technologies, compared them and selected the one we considered the most suitable according to our requirements (defined in WP2).

Considering supported data transmission rate and distance covered, we selected Wi-Fi as the ideal solution out of the available alternatives (listed here below, in Figure 5).

	Standard	Frequencies	Distance covered	Data transmission rate
Wi Fi	IEEE 802.11n IEEE 802.11ac	2,4 GHz 5 GHz	**< 50 m (indoor)** **< 100 m (outdoor)**	802.11n:150 Mbps up to 300 Mbps 802.11ac: **450** Mbps up to **1300** Mbps
Bluetooth	IEEE 802.15	2,4 GHz	< 100 m	1-3 Mbit/s
zigbee	Zigbee 3.0 (based on IEEE 802.15.4)	2,4 GHz	10-100 m	250 Kbit/s
Z-WAVE	Z-Wave Alliance	900 MHz	< 10 m (indoor) < 100 m (outdoor)	< 100 Kbit/s
LoRa	LoRaWAN	sub GHz	2-5 km (urban) 15 km (suburban) 30 km (rural)	< 50 Kbit/s

Figure 5 - Wireless technologies evaluated in Use Case 6

5.2.2 Use case diagram

The UML Use Case diagram reported in Figure 6 is related to an ideal scenario that can be summarized as follows: *"A person enters the totem area, goes in the totem proximity, stays correctly in front of its camera, speaks for at least 3 seconds, interacts with the totem as needed, goes away from its proximity, and exits from the totem area. Meanwhile, no other persons enter the totem area".*

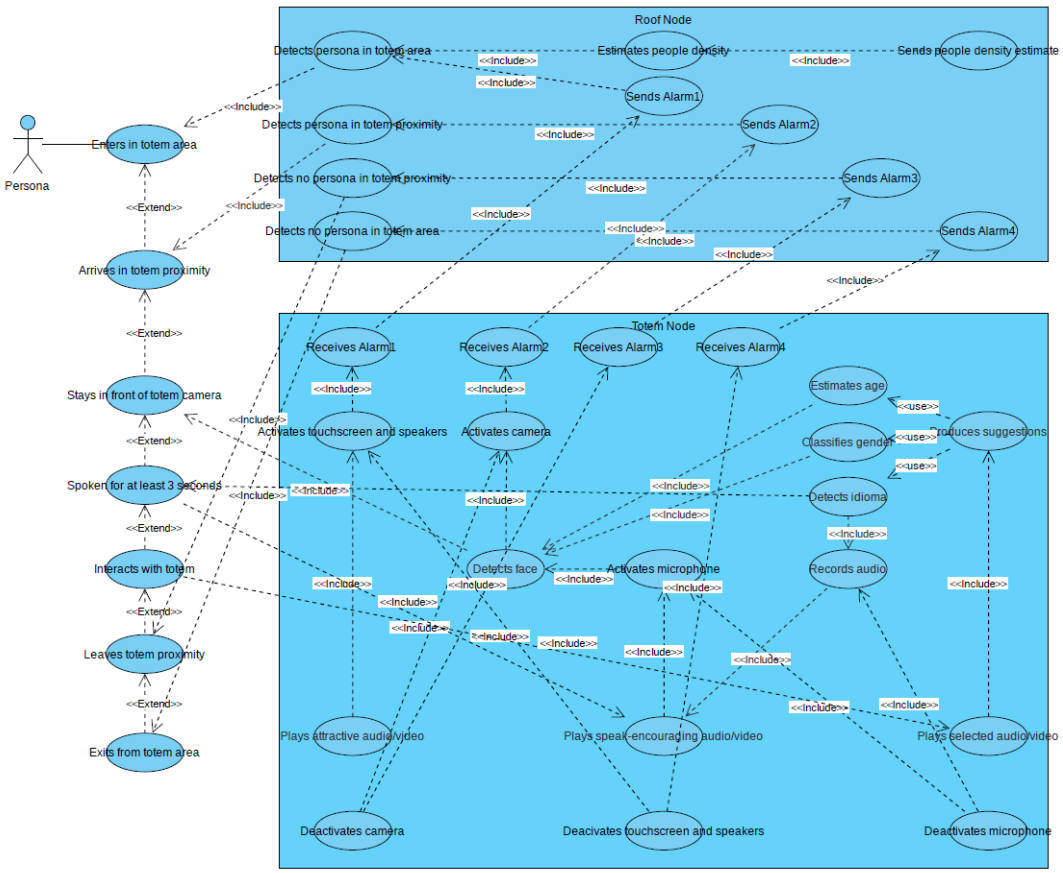

Figure 6 – UC6 diagram

Such an ideal scenario (described here in Figure 6) has been used to define the reference (HW/SW) system architecture (Figure 4) that will be considered for the system development and adapted to more complex scenarios. In particular, it will allow to evaluate the next steps with respect to the introduction of fully wireless connectivity and increasingly complex interactions with users and environment (e.g., more people in the totem area, multi-touchscreen totems). Moreover, it will be used to evaluate when the FRACTALITY features will be required to provide the processing power needed to manage more complex scenarios, i.e., to allow a totem to offload some of the processing to other nodes in the FRACTAL network.

The reference system architecture is composed of the following HW components:

- Roof Node
  - Xilinx Zynq Ultrascale+ connected via ethernet to IPCAM and MQTT Node
  - IPCAM (with quad-core ARM processor)
- Totem Node
  - Xilinx Zynq Ultrascale+ (VERSAL) connected via ethernet to IPCAM and MQTT Node and, via USB/HDMI to Touchscreen, Microphone and Speakers
  - IPCAM (with quad-core ARM processor)
  - Touchscreen

  - Speakers
  - Microphone
- MQTT Node
  - Raspberry-like (with quad-core ARM processor)

The reference system architecture is composed of the following SW components (the list also explicitly shows the mapping on the HW components):

- Roof Node
  - Xilinx Zynq Ultrascale+
    - Multi-core ARM
      - System SW: PetaLinux, Client MQTT
      - Application SW: Runtime Manager
    - FPGA (DPU)
      - Application SW: Density Estimator
  - IPCAM
    - System SW: OS
      - Application SW: People Detector
- Totem Node
  - IPCAM
    - System SW: OS
  - Xilinx Zynq Ultrascale+/VERSAL
    - Multi-core ARM
      - System SW: PetaLinux, Client MQTT
      - Application SW: Runtime Manager, Content Selector, Face Detector, Idiom Recognizer, Rule-Based Recommender
    - FPGA (DPU)
      - Application SW: Age Estimator, Gender Classifier
  - MQTT Node
    - System SW: Linux, Broker MQTT

In the considered scenario (Figure 6), and with respect to the reference system architecture (Figure 4), it is possible to describe with more details the main SW components.

Runtime Manager (RM) SW component is deployed on both the Roof and Totem Nodes. It is in charge of managing intra-node events and activities and inter-node communications. In particular, in the Roof Node, RM is in charge of dispatching alarms related to different events, while, in the Totem Node, it is in charge of receiving and managing such alarms other than managing local events and activities. Finally, Totem Node RM is also in charge of managing load balancing and possible computation offloading towards other nodes.

People Detector (PD) SW component, deployed on the Roof Node, triggers the main functionalities in the Totem Node. According to the position of people in the monitored area (detected by means of advanced and innovative Video Content Analysis), it generates different alarms that are sent to RM. In particular, as shown in Figure 6, the following alarms are generated:

- Alarm 1: generated when a person enters the totem area
- Alarm 2: generated when a person is detected in totem proximity
- Alarm 3: generated when no person is detected in totem proximity
- Alarm 4: generated when the totem area is empty

Since PD runs directly on the IPCAM composing the Roof Node, MQTT has been chosen as the protocol to communicate with the RM running on the other platform composing the Roof Node (i.e., Xilinx ZYNQ Ultrascale+). PD will send alarms to RM that will dispatch them to the Totem Node.

Density Estimator (**DE**) is another SW component deployed on the Roof Node. It strictly interacts with PD. In fact, given the people detection done by the PD SW component, it provides an estimation of the density of the people located in the totem area. Moreover, it could do also by itself people detection task (by processing video flows coming directly from the camera) and consequently compute the density estimation. Information generated by such components will be provided to the RM to be further sent, by means of MQTT, to a third-party external system for statistical analysis.

Face Detector (FD) SW component, deployed on the Xilinx ZYNQ Ultrascale+/VERSAL composing the Totem Node, receives a video stream from the Totem Node IPCAM to detect if a person is in front of it and, in that case, to detect faces. Detected face images are then sent to the Totem Node RM to be further dispatched to other SW components (i.e., Age Estimator and Gender Classifier).

Age Estimator (AE) SW component, deployed on the Xilinx ZYNQ Ultrascale+/VERSAL composing the Totem Node, receives detected face images from RM and estimates the age of the related persons. Such estimates are then provided back to RM to be further forwarded to the Rule-Based Recommender SW component (described below). AE is one of the SW components that can be offloaded to another node while RM performs load balancing.

Gender Classifier (GC) SW component, deployed on the Xilinx ZYNQ Ultrascale+/VERSAL composing the Totem Node, receives detected face images from RM and classifies the gender of the related persons. The results of such classifications are then provided back to RM to be further forwarded to the Rule-Based Recommender SW component (described below). GC is one of the SW components that can be offloaded to another node while RM performs load balancing.

Idiom Recognizer (IR) SW component, deployed on the Xilinx ZYNQ Ultrascale+/VERSAL composing the Totem Node, receives audio recordings by the Totem Node IPCAM to identify the idiom of the related persons. The results of such recognitions are then provided back to RM to be further forwarded to the Rule-Based Recommender SW component (described below). IR is one of the SW components that can be offloaded to another node while RM performs load balancing.

Rule-Based Recommender (RBR) SW component, deployed on the Xilinx ZYNQ Ultrascale+/VERSAL composing the Totem Node, exploits the information provided

by AE, GC, and IR, to suggest to the Content Selector SW component (described below) the most appropriate content to be shown on the Totem Node Touchscreen.

Content Selector (CS) SW component, deployed on the Xilinx ZYNQ Ultrascale+/VERSAL composing the Totem Node, simply shows on the Totem Node Touchscreen the content requested by RM or suggested by RBR.

5.2.3 Data interfaces

This chapter provides a description of UC6 data interfaces. In **Errore. L'origine riferimento non è stata trovata.** it provides a detailed overview of the interfaces between each UC component previously listed. Finally, it contains a detailed description of each interface, following the template provided at the very beginning of this document.

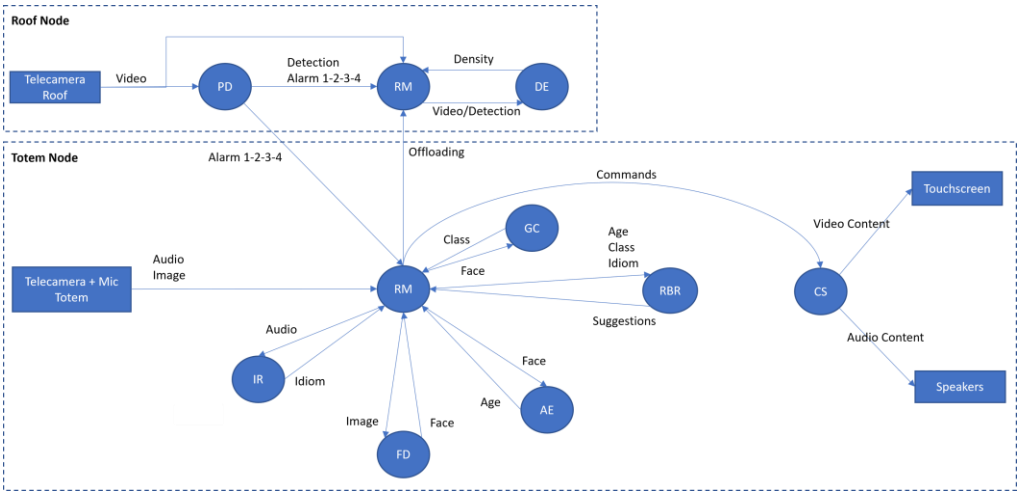

Figure 7 – UC6 data interfaces (focus on each component)

5.2.3.1 UC6_CMP_01 – Density Estimator

The **Density Estimator** (DE) component takes as input a video stream from a camera on the roof node. The aim of this component is to provide an estimation of the density of the people located in the totem area.

DE provides as output the density estimation and some performance metrics to the **Runtime Manager** (RM) component (**UC6_CMP_07**).

The RM component can re-configure the DE using the MQTT protocol to re-use information partially processed by the **People Detector** (PD) component (**UC_CMP_02***),* to save computational resources.

- UC_CMP_01-IN_01:
  - Source ID: UC6_CMP_07
  - Source Name: RM roof node
  - Data type: Video stream
  - Periodic / event-driven: Periodic (continuous stream)
  - Protocol definition (physical, logical / messages format):

        - Physical: Eth
        - Logical: RTSP protocol
        - Message format: H.264 1280x720 15fps
    - Other relevant info: this component allows to select INPUT 01 or INPUT 02 as source of data.
- UC_CMP_01-IN_02:
    - Source ID: UC6_CMP_07
    - Source Name: RM roof node
    - Data type: JSON containing detections
    - Periodic / event-driven: Periodic (continuous stream)
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared-memory
        - Logical: Bounding boxes
        - Message format: JSON
    - Other relevant info: this component allows to select INPUT 01 or INPUT 02 as source of data.

- UC_CMP_01-OUT_01:
    - Destination ID: UC6_CMP_07
    - Destination Name: RM roof node
    - Data type: JSON containing Density estimation and performance metrics.
    - Periodic / event-driven: Periodic (continuous stream)
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared-memory
        - Logical: Estimated density and performance metrics
        - Message format: JSON
    - Other relevant info: -

5.2.3.2 UC6_CMP_02 – People Detector

As detailed described here, this component collects as input video streams from a CCTV camera, processes them applying innovative AI algorithms and generates as output alarms and messages according to the information detected.

In more detail, the **UC6_CMP_02**, called **People Detector** and located in the roof node, has the objective to detect if a person is close to the totem area and to trigger alarms, to initialize the tasks of the totem node, communicating with the **Runtime Manager** (**UC6_CMP_07**). The **People Detector** component, will trigger different alarms based on where a person is located in the totem area. MQTT has been chosen as the transmission protocol. Moreover, it provides information about persons detected in the totem area also to another component of this use case, the **Density Estimator** (**UC6_CMP_01**).

- UC_CMP_02-IN_01:
    - Source ID: Cam_roof

  - Source Name: Roof node cam
  - Data type: Video stream
  - Periodic / event-driven: Periodic (continuos stream)
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: Rstp protocol
    - Message format: H.264 1280x720 15fps
  - Other relevant info: -

- UC_CMP_02-OUT_01:
  - Destination ID: UC6_CMP_07
  - Destination Name: RM roof and totem nodes
  - Data type: MQTT messages (alarms)
  - Periodic / event-driven: event-driven (triggered by vca)
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: MQTT protocol
    - Message format: JSON
  - Other relevant info: -
- UC_CMP_02-OUT_02:
  - Destination ID: UC6_CMP_07
  - Destination Name: RM roof node
  - Data type: MQTT messages (bounding boxes)
  - Periodic / event-driven: Periodic
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: MQTT protocol
    - Message format: JSON
  - Other relevant info: -

5.2.3.3 UC6_CMP_03 – Face Detector

The **Face Detector** (FD) component allows to obtain a frame centred on the user's face in front of the totem node. By using an image obtained by means of the camera positioned on the totem node as input, the component exploiting AI algorithms can automatically determine the faces inside the input image.

FD is triggered by the **Runtime Manager** component (**UC6_CMP_07**), based on events. Once the recognition process is completed, the detected face is stored inside a JPEG image that can be read by the **Runtime Manager** (**UC6_CMP_07**) using shared memory.

- UC_CMP_03-IN_01:
  - Source ID: UC6_CMP_07

- Source Name: RM Totem node
- Data type: Image frame
- Periodic / event-driven: event-driven
- Protocol definition (physical, logical / messages format):
  - Physical: Eth
  - Logical: RGB compressed image
  - Message format: JPEG image
- Other relevant info: -

- UC_CMP_03-OUT_01:
  - Destination ID: UC6_CMP_07
  - Destination Name: Runtime Manager
  - Data type: Image frame containing a face
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Shared memory
    - Logical: RGB compressed image
    - Message format: JPEG image
  - Other relevant info: the output of the FD component is written in a shared memory location that can be read by the RM component.

5.2.3.4 UC6_CMP_04 – Age Estimator

This component takes as input images of faces produced by the **Face Detector** component. By means of a neural network it estimates the age of the person and provides output to runtime manager. The output will be used by the Rule-Based Recommender component.

The **Age Estimator** is located on the totem node, but it can be also executed on roof node basing on the decision of Load Balancer component.

- UC_CMP_04-IN_01:
  - Source ID: UC6_CMP_07
  - Source Name: RM totem node
  - Data type: Image containing a face
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: shared-memory
    - Logical: RGB compressed image
    - Message format: JPEG image
  - Other relevant info: -

- UC_CMP_04-OUT_01:

- Destination ID: UC6_CMP_07
- Destination Name: RM totem node
- Data type: JSON
- Periodic / event-driven: event-driven
- Protocol definition (physical, logical / messages format):
  - Physical: Shared-Memory
  - Logical: Estimated Age
  - Message format: JSON Object
- Other relevant info: -

5.2.3.5 UC6_CMP_05 – Gender Classifier

The component takes as input images of faces produced by the **Face Detector** component. By means of a neural network it classifies the gender of the person and provides output to the **Runtime Manager**. The output will be used by the **Rule-Based Recommender** component.

The Gender Classifier is located on the totem node, but it can be also executed on roof node basing on the decision of **Load Balancer** component.

- UC_CMP_05-IN_01:
  - Source ID: UC6_CMP_07
  - Source Name: RM totem node
  - Data type: Image containing a face
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Shared memory
    - Logical: RGB compressed image
    - Message format: JPEG image
  - Other relevant info: -

- UC_CMP_05-OUT_01:
  - Destination ID: UC6_CMP_07
  - Destination Name: RM totem node
  - Data type: JSON
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Shared Memory
    - Logical: Classified Gender
    - Message format: JSON Object
  - Other relevant info: -

5.2.3.6 UC6_CMP_06 – Idiom Recognizer

The **Idiom Recognizer** is used to perform automatic language recognition based on speech processing. This component receives as input audio streams from a microphone, processes them by applying Machine Learning based Speech-to-Text (STT) solutions and hot-word detection to output the language of the current speaker.

In more detail, the **UC6_CMP_06**, called **Idiom Recognizer** and located in the roof node, has the objective to identify the language spoken by the current user and send this information to the **Runtime Manager** (**UC6_CMP_07**) in order to distribute the language info to the other components and automatically provide customized content to users in their spoken language (e.g., English, Italian, German, Spanish, French).

- UC_CMP_06-IN_01:
  - Source ID: UC6_CMP_07
  - Source Name: RM totem node
  - Data type: Audio stream
  - Periodic / event-driven: event-driven (people detection and audio acquisition)
  - Protocol definition (physical, logical / messages format):
    - Physical: Shared memory
    - Logical: system call
    - Message format: WAV
  - Other relevant info: -

- UC_CMP_06-OUT_01:
  - Destination ID: UC6_CMP_07
  - Destination Name: RM totem node
  - Data type: string (language)
  - Periodic / event-driven: event-driven (audio received and processed)
  - Protocol definition (physical, logical / messages format):
    - Physical: Shared memory
    - Logical: REST
    - Message format: JSON
  - Other relevant info: -

5.2.3.7 UC6_CMP_07 – Runtime Manager

The **Runtime Manager** component is used to manage tasks to be performed based on the input received and it is installed in each node. It also provides the load balancing feature thanks to the interface with the Load Balancer component.

The **Runtime Manager** needs to communicate with all the components installed in the node in order to coordinate them. The trigger that starts a task and its relative

scheduling is an alarm published on a MQTT by the **People Detector (UC6_CMP_02)**.

As mentioned before, it performs the load balancing, using results of the **Load Balancer** component, sending the payload to be computed to the **Runtime Manager** of another node which will return the result once the computation ends.

Since the **Runtime Manager** needs to communicate with different components, it has to handle different input and output data types.

- UC_CMP_07-IN_01:
  - Source ID: Cam_totem
  - Source Name: Totem node cam
  - Data type: Video stream
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: Video stream
    - Message format: RTSP stream
  - Other relevant info: -
- UC_CMP_07-IN_02:
  - Source ID: Cam_totem
  - Source Name: Totem node cam
  - Data type: Audio stream (containing customer voice)
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: Audio stream
    - Message format: RTSP stream
  - Other relevant info: -
- Data UC_CMP_07-IN_03:
  - Source ID: UC6_CMP_03
  - Source Name: Face detector
  - Data type: Image containing a face
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Shared memory
    - Logical: RGB compressed image
    - Message format: Image
  - Other relevant info: -
- UC_CMP_07-IN_04:
  - Source ID: UC6_CMP_04
  - Source Name: Age Estimator
  - Data type: JSON
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Shared-memory

        - Logical: Age estimated
        - Message format: JSON Object
    - Other relevant info: -
- UC_CMP_07-IN_05:
    - Source ID: UC6_CMP_05
    - Source Name: Gender Classifier
    - Data type: JSON
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared-memory
        - Logical: Gender classified
        - Message format: JSON Object
    - Other relevant info: -
- UC_CMP_07-IN_06:
    - Source ID: UC6_CMP_06
    - Source Name: Idiom Recognizer
    - Data type: Idiom recognized
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared memory
        - Logical: Idiom
        - IMessage format: JSON Object
    - Other relevant info: -
- UC_CMP_07-IN_07:
    - Source ID: UC6_CMP_08
    - Source Name: Rule Based Recommender
    - Data type: JSON
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared-memory
        - Logical: Content suggested
        - Message format: JSON Object
    - Other relevant info: -
- UC_CMP_07-IN_08:
    - Source ID: UC6_CMP_02
    - Source Name: People detector
    - Data type: MQTT messages (alarms)
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Eth
        - Logical: MQTT messages
        - Message format: JSON Object
    - Other relevant info: -

- UC_CMP_07-OUT_01:

    - Destination ID: UC6_CMP_07
    - Destination Name: RM roof node
    - Data type: JSON
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Eth
        - Logical: Offloading message and data (MQTT)
        - Message format: JSON format
    - Other relevant info: -
- UC_CMP_07-OUT_02:
    - Destination ID: CS
    - Destination Name: Content Selector
    - Data type: MQTT message
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared memory
        - Logical: Commands to display selected content
        - Message format: JSON Object
    - Other relevant info: -
- UC_CMP_07-OUT_03:
    - Destination ID: UC6_CMP_03
    - Destination Name: Face detector
    - Data type: Image
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared memory
        - Logical: Video stream
        - Message format: RTSP stream
    - Other relevant info: -
- UC_CMP_07-OUT_04:
    - Destination ID: UC6_CMP_04
    - Destination Name: Age Estimator
    - Data type: Image containing a face
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared-memory
        - Logical: RGB compressed image
        - Message format: JPEG image
    - Other relevant info: -
- UC_CMP_07-OUT_05:
    - Destination ID: UC6_CMP_05
    - Destination Name: Gender Classifier
    - Data type: Image containing a face
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared-memory

        - Logical: Gender classified
        - Message format: JPEG image
    - Other relevant info: -
- UC_CMP_07-OUT_06:
    - Destination ID: UC6_CMP_06
    - Destination Name: Idiom Recognizer
    - Data type: File audio (containing customer voice)
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared memory
        - Logical: File audio
        - Message format: WAV
    - Other relevant info: -
- UC_CMP_07-OUT_07:
    - Destination ID: UC6_CMP_08
    - Destination Name: Rule Based Recommender
    - Data type: JSON
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared-memory
        - Logical: Age, Gender, Idiom
        - Message format: JSON Object
    - Other relevant info: -

5.2.3.8 UC6_CMP_08 – Rule-based Recommender

The **Rule-based Recommender** system takes results of the other modules, aggregated as a JSON by the **Runtime Manager** and returns the decision about the displayed contents. The output, contained in a JSON is returned to the **Runtime Manager** to implement the corresponding decision.

- UC_CMP_08-IN_01:
    - Source ID: UC6_CMP_07
    - Source Name: RM totem node
    - Data type: JSON
    - Periodic / event-driven: event-driven
    - Protocol definition (physical, logical / messages format):
        - Physical: Shared memory
        - Logical: Age, Gender, Idiom
        - Message format: JSON Object
    - Other relevant info: -

- UC_CMP_08-OUT_01:
    - Destination ID: UC6_CMP_07

- Source Name: RM totem node
- Data type: JSON
- Periodic / event-driven: event-driven
- Protocol definition (physical, logical / messages format):
  - Physical: Shared memory
  - Logical: Suggestion
  - Message format: JSON Object
- Other relevant info: -

5.2.3.9 UC6_CMP_09 – Data Compressor

The **Data Compressor** component provides on-request compression and decompression functionalities for audio files. The component receives as input a .wav file and performs data compression via the LZW technique, returning a .lzw file. For decompression, the component takes as input a .lzw file and performs data decompression, returning a .wav file.

This component is a software library which shall be imported and used inside the **Runtime Manager (UC6_CMP_07)** when it is necessary to optimize the data rate for the transmission of an audio file.

- UC_CMP_09-IN_01:
  - Source ID: UC6_CMP_07
  - Source Name: Runtime Manager
  - Data type: Raw file
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: File
    - Logical: SW library
    - Message format: binary
  - Other relevant info: -

- UC_CMP_09-OUT_01:
  - Destination ID: UC6_CMP_07
  - Destination Name: Runtime Manager
  - Data type: Compressed file
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Shared Memory
    - Logical: Compressed file
    - Message format: binary
  - Other relevant info: -

6 VAL-UC7 Autonomous robot for implementing safe movements

6.1 Functional and non-functional requirements

6.1.1 Requirements from WP2

Req ID	Description
REQ_UC7_01	The edge node shall calculate nearest object distance from a cost map.
REQ_UC7_02	The edge node shall initiate emergency brake if detected object in inside defined area of the costmap.
REQ_UC7_03	The edge node shall calculate optimal trajectory based on the planned path and obstacles.
REQ_UC7_04	The edge node shall monitor metrics to initiate an emergency brake at connection loss or timeout.
REQ_UC7_05	The edge node shall be able to process real-time cost map data of UC7 at a rate up to 50 Hz.
REQ_UC7_06	The edge node shall provide an ethernet connection with TCP/UDP stack.
REQ_UC7_07	The edge node shall provide libraries for ROS2 support.
REQ_UC7_08	The edge node shall provide a node for diverse redundancy to avoid common cause faults.
REQ_UC7_09	The edge node shall support LEDEL library.
REQ_UC7_10	The edge node should support calculation of maximum timeframe for 10 Hz processing loop-rates.
REQ_UC7_11	The edge node shall provide a hardware accelerator supporting NN of UC7.

Table 9 - UC7 requirements from WP2

6.1.2 Requirements refinement during Use Cases specification

6.1.2.1 Requirements confirmed

All requirements defined in WP2 are confirmed.

6.1.2.2 Requirements removed

No requirements from WP2 were removed.

6.1.2.3 Requirements modified

No requirements from WP2 were modified.

6.1.2.4 Requirements added

Further requirements were derived from the SPIDER safety analysis according to ISO 26262, described in deliverable D4.4. Therefore, we analysed which of the resulting

safety goals are related to the functions from the implementation plan and not covered by the KPIs of the justification file.

Req ID	Description
REQ_UC7_12	Prevent no or wrong emergency stop activation when emergency stop is required in any driving situation.
REQ_UC7_13	Prevent wrong path (following of SPIDER in any driving situation).

Table 10 - UC7 requirements added

6.1.2.5 Consolidated list of requirements

Req ID	Description
REQ_UC7_01	The edge node shall calculate nearest object distance from a cost map.
REQ_UC7_02	The edge node shall initiate emergency brake if detected object in inside defined area of the costmap.
REQ_UC7_03	The edge node shall calculate optimal trajectory based on the planned path and obstacles.
REQ_UC7_04	The edge node shall monitor metrics to initiate an emergency brake at connection loss or timeout.
REQ_UC7_05	The edge node shall be able to process real-time cost map data of UC7 at a rate up to 50 Hz.
REQ_UC7_06	The edge node shall provide an ethernet connection with TCP/UDP stack.
REQ_UC7_07	The edge node shall provide libraries for ROS2 support.
REQ_UC7_08	The edge node shall provide a node for diverse redundancy to avoid common cause faults.
REQ_UC7_09	The edge node shall support LEDEL library.
REQ_UC7_10	The edge node should support calculation of maximum timeframe for 10 Hz processing loop-rates.
REQ_UC7_11	The edge node shall provide a hardware accelerator supporting NN of UC7.
REQ_UC7_12	Prevent no or wrong emergency stop activation when emergency stop is required in any driving situation.
REQ_UC7_13	Prevent wrong path (following of SPIDER in any driving situation).

Table 11 - UC7 consolidated list of requirements

6.2 Data interfaces

6.2.1 Use case Architecture and Components

UC7 is composed of two functions, the collision avoidance function, and the path tracking function, as shown in Figure 8. Both are implemented as FRACTAL nodes on the same hardware, based on NOEL-V. The nodes are using the ROS2 framework to communicate with the required sensors and SPIDER components running on different hardware.

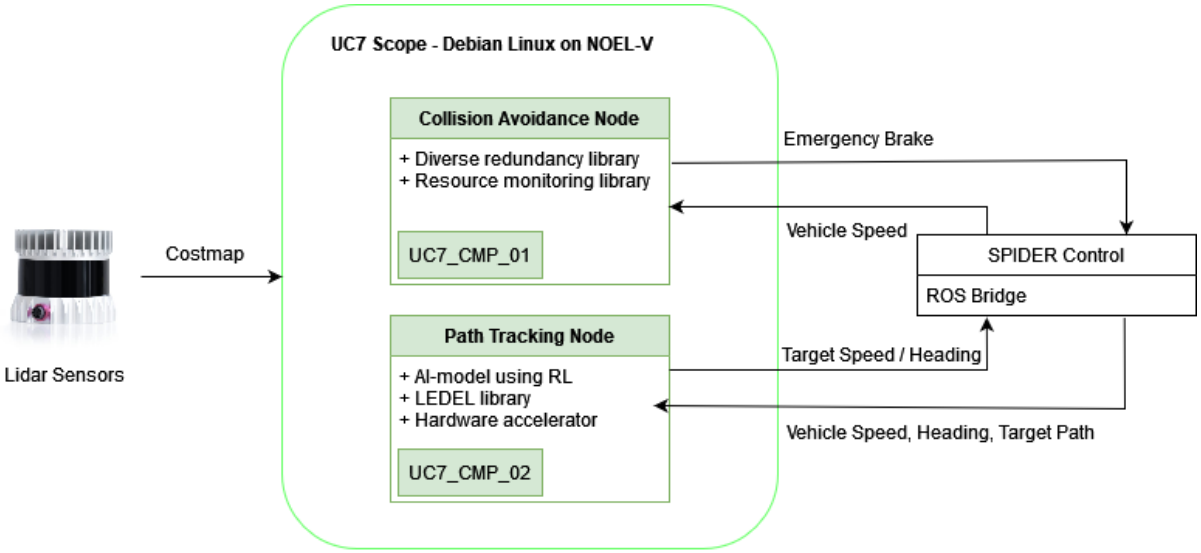

Figure 8 - UC7 architecture and components

6.2.1.1 UC7_CMP_01 – Collision Avoidance Node

The collision avoidance function is implemented as a FRACTAL node on the NOEL-V based platform. It communicates via ROS2 middleware with the required components from SPIDER robot. The node receives input from the lidar sensors, whereas the received sensor data are already pre-processed to costmap (or occupancy map) data. Further, the node receives the current vehicle speed from the SPIDER controllers. From this data, the function can calculate potential collisions and trigger an emergency brake.

- Data INPUT / UC7_IF_01:
  - Source Name: Costmap Generator
  - Data type: ROS2 message
  - Periodic / event-driven: periodic
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: TCP/IP
    - Message format: ROS2 message
  - Other relevant info: -
- Data INPUT / UC7_IF_02:
  - Source Name: Localization Module
  - Data type: ROS2 message
  - Periodic / event-driven: periodic
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: TCP/IP
    - Message format: ROS2 message
  - Other relevant info: -
- Data OUTPUT / UC7_IF_03:
  - Destination Name: Speed Limiter

- Data type: ROS2 message
- Periodic / event-driven: periodic, event-driven
- Protocol definition (physical, logical / messages format):
  - Physical: Eth
  - Logical: TCP/IP
  - Message format: ROS2 message
- Other relevant info: -

6.2.1.2 UC7_CMP_02 – Path Tracking Node

The path tracking function is implemented as a FRACTAL node on the NOEL-V based platform. It communicates via ROS2 middleware with the required components of the SPIDER robot. The node receives input from the lidar sensors, whereas the received sensor data are already pre-processed to costmap (or occupancy map) data. Further the node receives the current vehicle speed, heading, and the target path from the SPIDER controllers. From this data, the function can calculate the best trajectory and replying with the target speed and orientation to the SPIDER controllers.

- Data INPUT / UC7_IF_01:
  - Source Name: Costmap Generator
  - Data type: ROS2 message
  - Periodic / event-driven: periodic
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: TCP/IP
    - Message format: ROS2 message
  - Other relevant info: -
- Data INPUT / UC7_IF_02:
  - Source Name: Localization Module
  - Data type: ROS2 message
  - Periodic / event-driven: periodic
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: TCP/IP
    - Message format: ROS2 message
  - Other relevant info: -
- Data INPUT / UC7_IF_04:
  - Source Name: Path Planner
  - Data type: ROS2 message
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: TCP/IP
    - Message format: ROS2 message
  - Other relevant info: -
- Data OUTPUT / UC7_IF_05:

- Destination Name: Speed Limiter
- Data type: ROS2 message
- Periodic / event-driven: periodic, event-driven
- Protocol definition (physical, logical / messages format):
  - Physical: Eth
  - Logical: TCP/IP
  - Message format: ROS2 message
- Other relevant info: -

6.2.2 Use case diagram

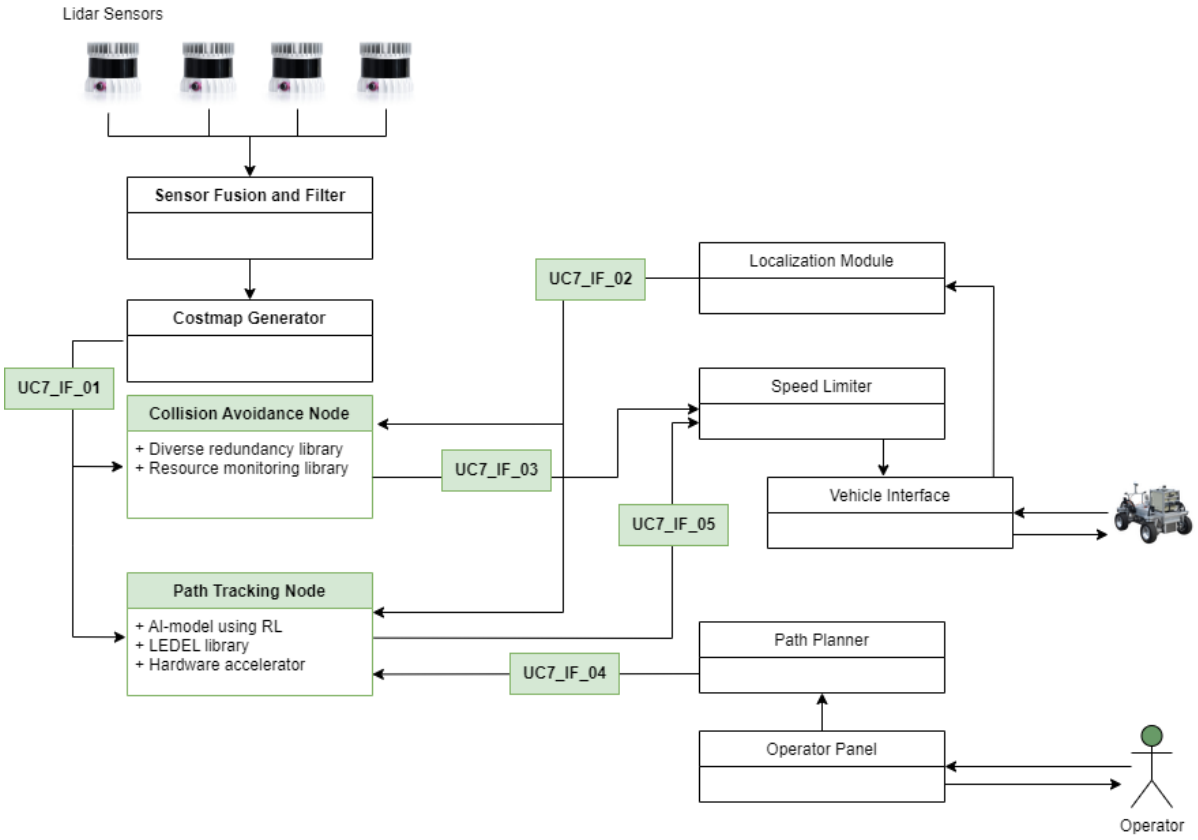

Figure 9 - UC7 use case diagram with data interfaces

6.2.3 Data interfaces

Figure 9 shows the data interfaces for UC7. All the interfaces are implemented via the ROS2 middleware[1], which is an implementation of the Data Distribution Service (DDS)[2] standard. The interfaces are identified by a topic name and defined by a message type.

[1] https://docs.ros.org/en/foxy/Concepts/About-Middleware-Implementations.html
[2] https://www.dds-foundation.org/omg-dds-standard/

6.2.3.1 UC7_IF_01

The interface uses a ROS2 standard message.

Topic Name: /spider/collision_avoidance/costmap/occupancy_grid
Message Type: nav_msgs/msg/OccupancyGrid
Message Definition: https://docs.ros2.org/latest/api/nav_msgs/msg/OccupancyGrid.html

Table 12 – UC7_IF_01 details

6.2.3.2 UC7_IF_02

The interface uses a ROS2 standard message.

Topic Name: /spider/localization/map, /spider/localization/odom
Message Type: nav_msgs/msg/Odometry
Message Definition: https://docs.ros2.org/latest/api/nav_msgs/msg/Odometry.html

Table 13 – UC7_IF_02 details

6.2.3.3 UC7_IF_03

Topic Name: /spider/estop_heartbeat		
Message Type: spider_vehicle_msgs/msg/EStopHeartbeat		
Field	**Description**	**Length**
Originator	Identifier of message originator	Unbounded
Stamp	Timestamp of sending	Time
Type	0…STANDBY 1…EMERGENCY_BRAKE	uint8

Table 14 – UC7_IF_03 details

6.2.3.4 UC7_IF_04

Topic Name: /spider/estop_heartbeat		
Message Type: spider_path_tracking_msgs/msg/Path		
Field	**Description**	**Length**
Header	ROS Header message	std_msgs/msg/Header
Waypoints	List of waypoints	spider_path_tracking_msgs/msg/Waypoint[]

Table 15 – UC7_IF_04 details (1/2)

Message Type: spider_path_tracking_msgs/msg/Waypoint		
Field	**Description**	**Length**

X	X coordinate in local coordinate system	float64
Y	Y coordinate in local coordinate system	float64
Vel	Target velocity at (x/y)	float64
Rot	Unused	float64
Yaw	Target orientation at (x/y)	float64
Rc	Unused	float46
Accel	Unused	float64

Table 16 – UC7_IF_04 details (2/2)

6.2.3.5 UC7_IF_05

The interface uses a ROS2 standard message.

Topic Name: /spider/cmd/auto
Message Type: geometry_msgs/msg/Twist
Message Definition: https://docs.ros2.org/latest/api/geometry_msgs/msg/Twist.html

Table 17 – UC7_IF_05 details

7 VAL-UC8 Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse

7.1 Functional and non-functional requirements

7.1.1 Requirements from WP2

Req ID	Description
REQ_UC8_##	Safety Certification for Black-Channel communication (ASIL 3, ISO 26262) In Order to allow the storage device to be ASIL 3 compliant, the communication channel needs to be certified
REQ_UC8_##	Real-Time capabilities The FRACTAL Node should provide real time capabilities in order to correctly control the storage devices with a RT-Patch enabled Linux OS.
REQ_UC8_##	Pathfinding To allow storage and path improvements, the AI of the FRACTAL node should support pathfinding to allow enable the nodes to find the best available shuttle for tasks
REQ_UC8_##	Machine learning To allow predictive maintenance features to be developed, machine learning is required to predict failures of certain parts and devices
REQ_UC8_##	Wireless Communication To allow communication with other FRACTAL nodes and external systems, WiFi Communication is required with a minimum Bandwidth of 300 Mbit/s. Ad-Hoc (mesh) and Access-Point based connections are necessary
REQ_UC8_##	Diagnostic protocol to shutdown device on communication loss
REQ_UC8_##	2 1000Mbit/s Ethernet interfaces
REQ_UC8_##	Localization features (Localization of device inside of the warehouse)
REQ_UC8_##	• The node shall handle multi-threading applications. • The node shall have at least 800 MHz on each core. • The node shall provide at least 4GB DDR RAM. • The node shall provide at least 32 GB eMMC or similar memory
REQ_UC8_##	The FRACTAL node shall provide an EtherCAT stack on 1 of 2 Ethernet interfaces
REQ_UC8_##	The FRACTAL node shall provide a ProfiNET Master stack
REQ_UC8_##	Linux with RT Patch as Operating System
REQ_UC8_##	Gyroscope
REQ_UC8_##	1 CANOpen interface (D-SUB9)
REQ_UC8_##	Serial TTY interface
REQ_UC8_##	1 USB3 Port

Table 18 – UC8 requirements defined in WP2

7.1.2 Requirements refinement during Use Cases specification

7.1.2.1 Requirements confirmed

All requirements from WP2 were changed or removed as the initial requirements specifications were in an early stage of integration. With internal engineering effort and technical meetings/workshops on- and offline in the project, it was clear to modify the first requirements.

7.1.2.2 Requirements removed

The listed requirements were removed during use case specification, as the most requirements are depending on own realization effort and not in context of FRACTAL.

Req ID	Description
REQ_UC8_##	Diagnostic protocol to shutdown device on communication loss
REQ_UC8_##	2 1000Mbit/s Ethernet interfaces
REQ_UC8_##	Localization features (Localization of device inside of the warehouse)
REQ_UC8_##	The FRACTAL node shall provide an EtherCAT stack on 1 of 2 Ethernet interfaces
REQ_UC8_##	The FRACTAL node shall provide a ProfiNET Master stack
REQ_UC8_##	To allow predictive maintenance features to be developed, machine learning is required to predict failures of certain parts and devices
REQ_UC8_##	Gyroscope
REQ_UC8_##	Serial TTY interface
REQ_UC8_##	1 USB3 Port

Table 19 - Removed requirements UC8

7.1.2.3 Requirements modified

Some requirements need to be modified to define them more precisely and provide afterwards a clear separation between topics, e.g., AI models, connectivity.

From: REQ_UC8_##	• The node shall handle multi-threading applications. • The node shall have at least 800 MHz on each core. • The node shall provide at least 4GB DDR RAM. • The node shall provide at least 32 GB eMMC or similar memory
To: REQ_UC8_01	The edge node should have followed hardware specification: - at least 2 cores @ 800 MHz - at least 4 GB RAM - at least eMMC Memory or similar.
From: REQ_UC8_##	Wireless Communication To allow communication with other FRACTAL nodes and external systems, WiFi Communication is required with a minimum Bandwidth of 300 Mbit/s. Ad-Hoc (mesh) and Access-Point based connections are necessary

From: REQ_UC8_##	1 CANOpen interface (D-SUB9)
To: REQ_UC8_02	These communication protocols shall be used from Linux OS: - MQTT over Wi-Fi mesh network for communication between nodes - CAN Bus for internal communication.
From: REQ_UC8_##	Pathfinding To allow storage and path improvements, the AI of the FRACTAL node should support pathfinding to allow enable the nodes to find the best available shuttle for tasks
To: REQ_UC8_05	The edge node shall be able to use an adaptive orchestrator (scheduler) for storing strategies and optimized pathfinding for each shuttle depending on material (weight, type), frequency of requests, division of same type in different levels for alternative access/ faster access on big order amount.
To: REQ_UC8_06	The edge node shall offer optimized pathfinding: Improving path of the shuttles, for different scenarios; obstacle in same layer; malfunction of a shuttle; avoiding crossing in same level.
From: REQ_UC8_##	Real-Time capabilities The FRACTAL Node should provide real time capabilities to correctly control the storage devices with a RT-Patch enabled Linux OS.
To: REQ_UC8_07	The node shall feature Linux operating system with real time capability (e.g., time-triggered communication capabilities).
From: REQ_UC8_##	Safety Certification for Black-Channel communication (ASIL 3, ISO 26262) In Order to allow the storage device to be ASIL 3 compliant, the communication channel needs to be certified
To: REQ_UC8_08	Safety wireless communication should be over a black channel (ASIL 3, ISO 26262) between nodes.

Table 20 - Modified requirements UC8

7.1.2.4 Requirements added

REQ_UC8_03	The edge node shall provide enough interfaces for two cameras.
REQ_UC8_04	The edge node shall be capable to detect objects (human body and other obstacles) from video input stream of the provided cameras and evaluate the detected object to generate a safe output, if the obstacle is in a defined range of the shuttle.
REQ_UC8_09	For the edge nodes a cross compiler shall be available to port control software.
REQ_UC8_10	The edge node shall support libraries, like TensorFlow/ Keras.

Table 21 - Added requirements UC8

7.1.2.5 Consolidated list of requirements

Req ID	Description
REQ_UC8_01	The edge node should have following hardware specification: - at least 2 cores @ 800 MHz - at least 4 GB RAM - at least eMMC Memory or similar.

REQ_UC8_02	These communication protocols shall be used from Linux OS: - MQTT over Wi-Fi mesh network for communication between nodes - CAN Bus for internal communication.
REQ_UC8_03	The edge node shall provide enough interfaces for two cameras.
REQ_UC8_04	The edge node shall be capable to detect objects (human body and other obstacles) from video input stream of the provided cameras and evaluate the detected object to generate a safe output, if the obstacle is in a defined range of the shuttle.
REQ_UC8_05	The edge node shall be able to use an adaptive orchestrator (scheduler) for storing strategies and optimized pathfinding for each shuttle depending on material (weight, type), frequency of requests, division of same type in different levels for alternative access/ faster access on big order amount.
REQ_UC8_06	The edge node shall offer optimized pathfinding: Improving path of the shuttles, for different scenarios; obstacle in same layer; malfunction of a shuttle; avoiding crossing in same level.
REQ_UC8_07	The node shall feature Linux operating system with real time capability (e.g., time-triggered communication capabilities).
REQ_UC8_08	Safety wireless communication should be over a black channel (ASIL 3, ISO 26262) between nodes.
REQ_UC8_09	For the edge nodes a cross compiler shall be available to port control software.
REQ_UC8_10	The edge node shall support libraries, like TensorFlow/ Keras.

Table 22 - Consolidated requirements UC8

7.2 Data interfaces

7.2.1 Use case Architecture and Components

The test setup architecture consists of three FRACTAL edge nodes, what typically depends on the number of shuttles in the system. More particularly in this case, one FRACTAL edge node is used to control two elevators (Versal) of the system and two more for each shuttle (Kria), like shown in Figure 10. Additionally, the FRACTAL cloud solution is used as fleet management system, where AI models and control services are stored and managed per project.

The communication between the nodes is realized by MQTT. The broker is located in the Versal node and every other edge node in the swarm subscribes to the corresponding swarm broker. Deploying of prepared models or control services from the cloud will be initiated by an implemented update routine over a secure HTTPS connection.

Physically, the edge nodes implemented in the shuttles are connected via Wi-Fi (red dotted lines) and the Elevator node via Ethernet (green dashed lines). The interconnection is established by the combination of an access point and a switch.

Internal connection, more accurate shuttle and elevator control services use CAN-Bus connectivity to control drives, as well as sensors and contactors etc.

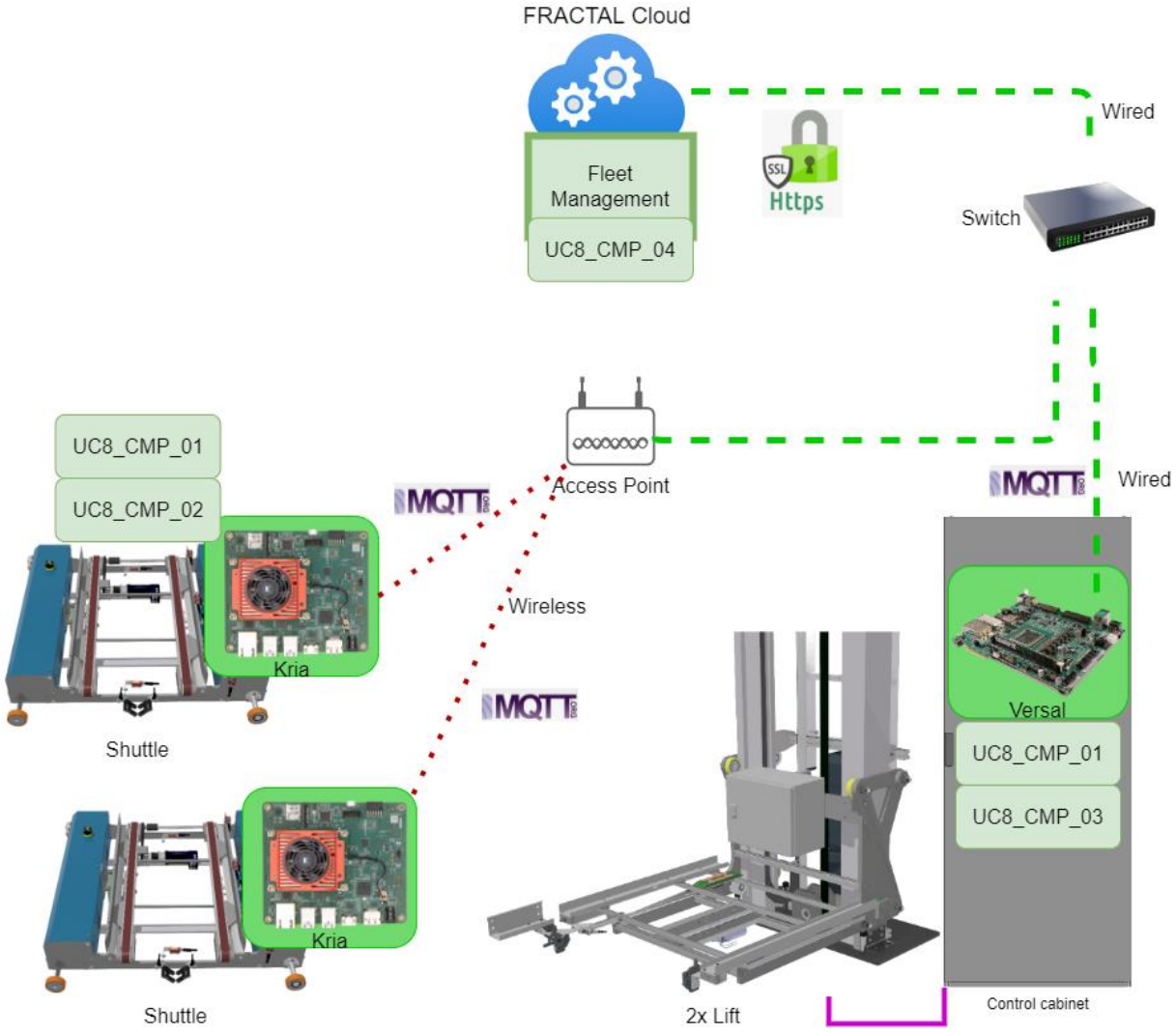

Figure 10 - Architecture of the test setup UC8

The list of UC components (the location can be found in Figure 10):

- UC8_CMP_01: Hardware design with CAN Bus connectivity
  FPGA Hardware design ready with communication interfaces between components like IOs and motors, but also FRACTAL components.
- UC8_CMP_02: Evaluation of object detection
  Safety relevant evaluation logic and connection to the safety plc
- UC8_CMP_03: AI accelerated orchestrator/ scheduler
  Warehouse optimization and pathfinding based on the metascheduler
- UC8_CMP_04: Cloud service orchestration
  Fleet management system

7.2.2 Use case diagram

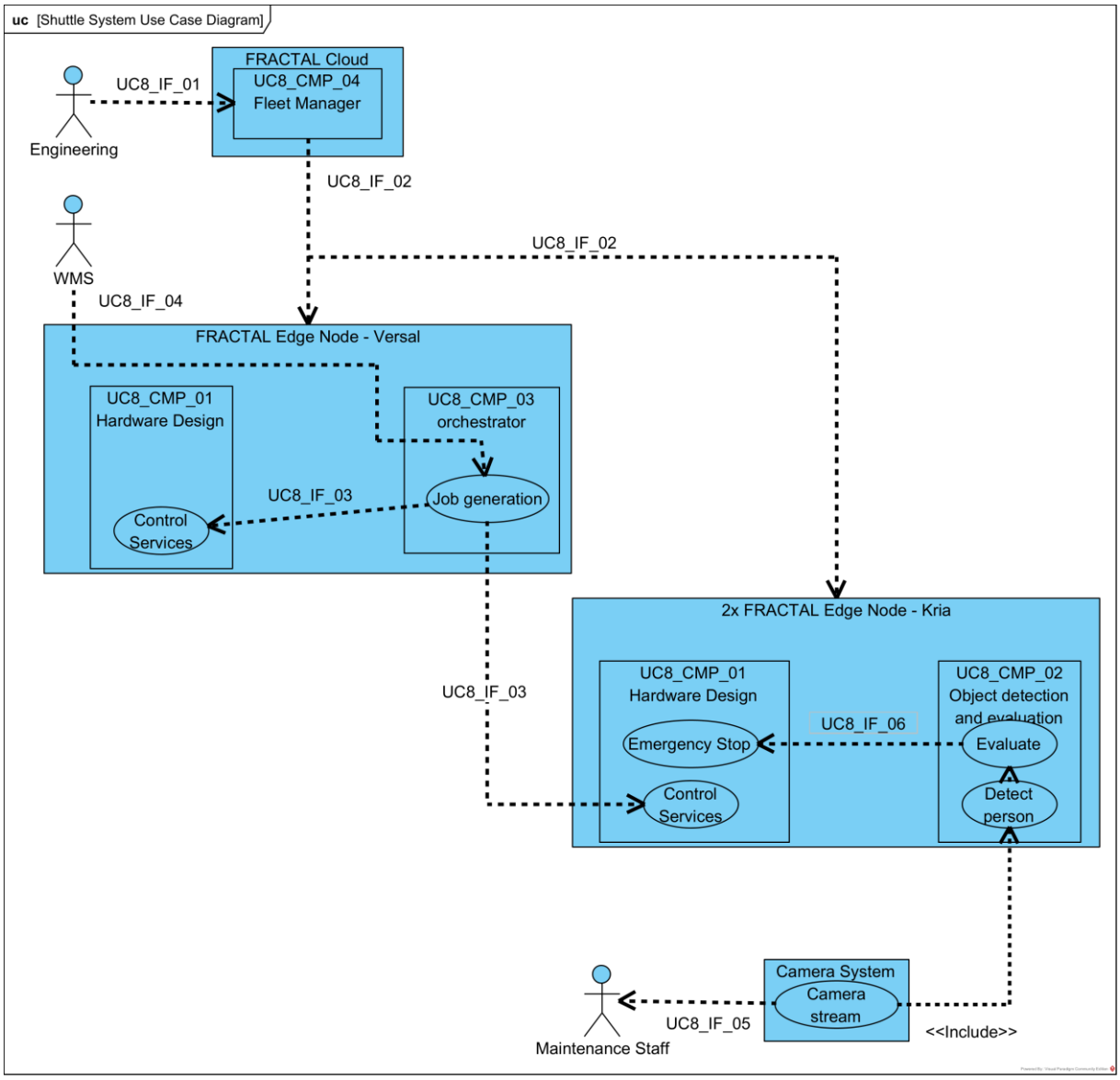

Figure 11 - Use case diagram UC8

Figure 11 shows the basic principle of interconnection between the use case components with defined interfaces. For example, the engineering team maintains the data in the FRACTAL cloud, but also triggers the update process (UC8_IF_01) for their responsible project (UC8_IF_02). The job management of the orchestrator receives the requests from the warehouse management system (UC8_IF_03) and generates the required message for each swarm member (UC8_IF_03).

Regarding the camera evaluation, the implemented camera system has an open stream over USB (UC8_IF_05) to the person detection model. The processed data is evaluated in the next step and connected to the emergency stop circuit in the shuttle

(UC8_IF_06). When the evaluation output is out of the defined rules, the safety plc triggers the emergency stop.

7.2.3 Data interfaces

7.2.3.1 UC8_CMP_01: Hardware design with CAN Bus connectivity

The hardware design contains FRACTAL and use case specific components. In consideration of the use case only, the interfaces are required to process job messages and handle the internal communication.

- Data INPUT 01 / UC8_IF_02:
  - Source Name: Image repository (UC8_CMP_04)
  - Data type: Docker image
  - Periodic / event-driven: event-driven (manual)
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth, Wi-Fi
    - Logical: HTTPS/ SSH
    - Message format: docker pull
  - Other relevant info: Controlled update process of models and control services.

- Data INPUT 02 / UC8_IF_03:
  - Source Name: Job message
  - Data type: MQTT message
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth, Wi-Fi
    - Logical: MQTT protocol
    - Message format: JSON
  - Other relevant info: Generated job for each swarm member (elevators and shuttles per swarm).
- Data OUTPUT 01 / UC8_IF_03:
  - Source Name: Status message
  - Data type: MQTT message
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth, Wi-Fi
    - Logical: MQTT protocol
    - Message format: JSON
  - Other relevant info: Logging of status data from each swarm member. E.g., position, member status, error messages.

7.2.3.2 UC8_CMP_02: Evaluation of object detection

Implemented person detection on the shuttle nodes with integration in the safety plc.

- Data INPUT 03 / UC8_IF_05:
  - Source Name: Camera stream
  - Data type: Video Stream
  - Periodic / event-driven: periodic (continues stream)
  - Protocol definition (physical, logical / messages format):
    - Physical: USB-C 3.1 Gen 1
    - Logical: RAW RGB colour
    - Message format: YUY2 [16 Bit] 1280x720 @ 15 fps

- Data INPUT 04 / UC8_IF_05:
  - Source Name: Camera stream
  - Data type: Video Stream
  - Periodic / event-driven: periodic (continues stream)
  - Protocol definition (physical, logical / messages format):
    - Physical: USB-C 3.1 Gen 1
    - Logical: RAW Depth
    - Message format: Z [16 Bit] 848x480 @ 10 fps

- Data OUTPUT 02 / UC8_IF_06:
  - Source Name: Safety signal
  - Data type: Safe output
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: hardwired
    - Logical: Clocked signal
    - Message format: none

7.2.3.3 UC8_CMP_03: AI accelerated orchestrator/ scheduler

Orchestrator model for shuttle swarm orchestration.

- Data INPUT 05 / UC_IF_04:
  - Source Name: Fulfilment request
  - Data type: MQTT message
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: MQTT protocol
    - Message format: JSON
  - Other relevant info: Incoming order from warehouse management system.

- Data OUTPUT 03 / UC_IF_04:
  - Source Name: Job message
  - Data type: MQTT message
  - Periodic / event-driven: event-driven
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: MQTT protocol
    - Message format: JSON
  - Other relevant info: Scheduled jobs for all swarm members.

7.2.3.4 UC8_CMP_04: Cloud service orchestration

Fleet management system from the cloud, with model preparation and versioning functionalities per project.

- Data INPUT 06 / UC8_IF_01:
  - Source Name: Image repository
  - Data type: Docker image
  - Periodic / event-driven: event-driven (manual)
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth
    - Logical: HTTPS/ SSH
    - Message format: docker build
  - Other relevant info: Management of AI models, data sets and control services.

- Data OUTPUT 04 / UC8_IF_02:
  - Source Name: Image repository (UC8_CMP_04)
  - Data type: Docker image
  - Periodic / event-driven: event-driven (manual)
  - Protocol definition (physical, logical / messages format):
    - Physical: Eth, Wi-Fi
    - Logical: HTTPS/ SSH
    - Message format: Bash script
  - Other relevant info: Outgoing update process.

8 List of figures

9 List of tables