Ref. Ares(2022)720044 - 31/01/2022

D1.4 – SECOND VERSION OF CPSOSAWARE SYSTEM ARCHITECTURE

Authors	Pavlos Kosmides (CTL), Panagiotis Mitzias (CTL), Efstratios Kontopoulos (CTL), Eleni Adamopoulou (CTL)
Work Package	WP1 – Requirements, Use Cases, Specifications and Architecture

Abstract

This document presents an updated report on technical specifications, system requirements and architecture of the CPSoSaware system. It introduces the applied platform specification methodology, which defines a set of distinct viewpoints for the system design: context, components, requirements, distribution, and realization views. Subsequently, the document presents the latest status of three of these views that are considered pertinent to this version of the architecture definition. Firstly, it reports the updated system decomposition to technical components with a focus on dependencies and interfaces among the latter. Secondly, the document lists the system requirements, serving as reference for their fulfilment status, priority, and monitoring. Lastly, the distribution of technical components to CPSoSaware use cases and architectural blocks is demonstrated.

Funded by the Horizon 2020 Framework Programme
of the European Union

Deliverable Information

Work Package	WP1 – Requirements, Use Cases, Specifications and Architecture
Task	T1.3 – CPSoSaware System Specifications and Architecture
Deliverable title	Second Version of CPSoSaware System Architecture
Dissemination Level	Public
Status	F
Version Number	1.0
Due date	31/12/2021

Project Information

Project start and duration	01/01/2020 – 31/12/2022, 36 months
Project Coordinator	Industrial Systems Institute, ATHENA Research and Innovation Center 26504, Rio-Patras, Greece
Partners	1. ATHINA-EREVNITIKO KENTRO KAINOTOMIAS STIS TECHNOLOGIES TIS PLIROFORIAS, TON EPIKOINONION KAI TIS GNOSIS (ISI) * Coordinator 2. FUNDACIO PRIVADA I2CAT, INTERNET I INNOVACIO DIGITAL A CATALUNYA (I2CAT), 3. IBM ISRAEL - SCIENCE AND TECHNOLOGY LTD (IBM ISRAEL 4. ATOS SPAIN SA (ATOS), 5. PANASONIC AUTOMOTIVE SYSTEMS EUROPE GMBH (PASEU) 6. EIGHT BELLS LTD (8BELLS) 7. UNIVERSITA DELLA SVIZZERA ITALIANA (USI), 8. TAMPEREEN KORKEAKOULUSAATIO SR (TAU) 9. UNIVERSITY OF PELOPONNESE (UoP) 10. CATALINK LIMITED (CATALINK) 11. ROBOTEC.AI SPOLKA Z OGRANICZONA ODPOWIEDZIALNOSCIA (RTC) 12. CENTRO RICERCHE FIAT SCPA (CRF) 13. PANEPISTIMIO PATRON (UPAT)
Website	www.cpsosaware.eu

Control Sheet

Version	Date	Summary of changes	Author
0.1	13/12/2021	Document outline created	CTL
0.2	17/12/2021	v1 of Chapters 3, 4 and 5	CTL
0.3	04/01/2022	v2 of Chapters 3, 4 and 5	CTL
0.4	05/01/2022	Authored Chapter 2	CTL
0.5	06/01/2022	Added executive summary & introduction	CTL
0.6	07/01/2022	v2 of Chapter 2	CTL
0.7	11/01/2022	Integrated partner contributions	CTL
0.8	13/01/2022	Authored Abstract and Chapter 6	CTL
0.9	17/01/2022	Submitted for internal review	CTL
1.0	31/01/2022	Final version submitted to EC	CTL

	Name
Prepared by	CTL
Reviewed by	Aris Lalos (ISI), Antonio Álvarez Romero (ATOS)
Authorised by	CTL

Date	Recipient
17/01/2022	Project Consortium
31/01/2022	European Commission

Table of Contents

List of Figures

List of Tables

Executive Summary

This document constitutes D1.4 “Second Version of CPSoSaware System Architecture” and reports on the outcomes of the second phase of Task 1.3 during the second year of the project. To this end, D1.4 builds upon the information reported in the preceding D1.3 “Preliminary Version of CPSoSaware System Architecture” to describe the progress in architecture definition using an agile system specification methodology. According to the latter, a series of distinct viewpoints has been defined for the decomposition of the system to technical components, the documentation of dependencies and interfaces, the recording and monitoring of system requirements, and the distribution of modules within the system. The generated views, that are presented in this document and constitute the main outputs of this deliverable, are meant to facilitate the system interpretation, implementation, extensibility, and maintainability.

1 Introduction

In the context of system design, **architecture** is defined as the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [1].

This technical report builds upon the documentation presented in D1.3 “Preliminary Version of CPSoSaware System Architecture” to elaborate on the headway of the architecture definition within the CPSoSaware project. A novel system specification methodology has been applied, that defines a set of views and prescribes how to use these in the architectural description to facilitate the system interpretation, implementation, extensibility, and maintainability.

1.1 Document Structure

The rest of the document is structured as follows:

- **Chapter 2** is an introduction to the platform specification methodology that has been adopted for the definition of architectural views;
- **Chapter 3** describes the component view, containing the system decomposition to technical components and dependencies/interfaces between them;
- **Chapter 4** presents the updated requirement view, including monitoring methodology and a report on the status of all system requirements;
- **Chapter 5** presents the distribution view of technical components to CPSoSaware use cases and architectural layers;
- **Chapter 6** concludes the document with some final remarks and directions for the next steps.

1.2 Definitions and Acronyms

Below is a list of the most relevant acronyms used in the document together with their recurring definitions:

Acronym / Term	Definition
AR	Augmented Reality
ASIP	Application-specific Instruction-set Processor
CL	Cooperative Localization
CNN	Convolutional Neural Network
CPS	Cyber-Physical System
CPSoS	Cyber-Physical System of Systems
CSV	Comma-separated Values

DCNN	Deep Convolutional Neural Network
DMS	Driver Monitoring System
DoF	Depth of Field
FPGA	Field-Programmable Gate Array
HLS	High-Level Synthesis
HW	Hardware
LiDAR	Light Detection And Ranging
MQTT	Message Queuing Telemetry Transport
OWL	Web Ontology Language
PoCL	Portable Computing Language
RDF	Resource Description Framework
RTL	Register Transfer Level
SHACL	Shapes Constraint Language
SRMM	Security Runtime Monitoring and Management
SW	Software
TCE	TTA-Based Co-design Environment
UML	Unified Modeling Language
XR	Extended Reality
XRT	Xilinx Run-time

2 CPSoSaware Platform Specification Methodology

For specifying v2 of the CPSoSaware platform architecture, we relied on the well-known **ARCADE framework**[1], a domain and technology-independent architectural description framework for software intensive systems. This chapter briefly presents the principles underlying ARCADE, focusing on the distinct viewpoints it defines for describing software system architectures.

2.1 Viewpoints and Views

The core of the ARCADE framework is the specification of **a set of viewpoints** for describing the software system architecture. Each viewpoint defines **how a specific view of the target system shall be described**, while accompanying diagrams (typically in UML) better illustrate the respective operations and interactions. In a nutshell, viewpoints are used to create a view, and each view consists of one or more models that specify different aspects related to the structure and behaviour of a target system.

ARCADE defines the following five viewpoints:

- **Context Viewpoint**: The context viewpoint defines the environment of the system, documenting what the target system is intended to do in its environment and containing the specification of use case scenarios and elicited requirements. The CPSoSaware context viewpoint has already been described in D1.2 “Requirements and Use Cases” (delivered in M15) and will not be further elaborated in this document.
- **Component Viewpoint**: The component viewpoint specifies the decomposition of the system into components (i.e., subsystems and information objects) and describes their interactions and interfaces. The CPSoSaware component viewpoint is presented in the next chapter.
- **Requirement Viewpoint**: The requirements viewpoint documents all requirements related to the target system, with the aim to verify that the latter is indeed capable to perform its initially intended tasks. Functional requirements for CPSoSaware were documented in D1.2, while technical requirements are presented in Chapter 4.
- **Distribution Viewpoint**: The logical distribution of components within the system architecture is the topic of the distribution viewpoint. The CPSoSaware distribution viewpoint is presented in Chapter 5.
- **Realisation Viewpoint**: The realisation viewpoint is aimed at describing the implementation of the subsystems, along with their deployment. Potential constraints regarding the implementation and/or deployment of the target system’s components should also be documented. This viewpoint is omitted from this deliverable and will be presented in the final iteration of the CPSoSaware architecture, D1.5, which is due M36.

2.2 Process Model

Figure 1 displays the high-level process model for developing the different views in the ARCADE framework. The context view is the more “high-level” one and should be documented first, followed by the “mid-level” views of requirements, component, and distribution. Finally, the “lowest-level” realization view should be documented.

[1] http://arcade-framework.org/

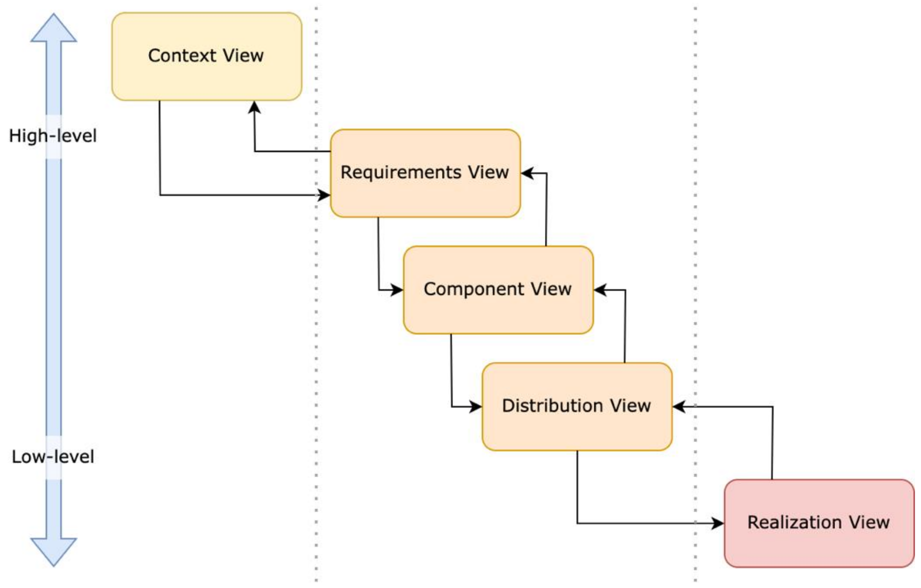

Figure 1 - Process for documenting the different views in ARCADE.

This is not a strictly sequential process in the sense that a view does not need to be finished before the next view can be documented. Higher views can also be partially finished in the first iteration, before a lower-level view can be initiated; upper views can then be revisited and refined. However, a hard constraint is that a lower-level view should not document elements that have not yet been included in higher levels.

This approach is fully adopted within CPSoSaware as well, with three iterations of the system architecture (v1, v2, final) being documented in three respective deliverables (D1.3, D1.4, D1.5), with each iteration refining and extending the previous one.

3 Component View

In this section, the component view is reported, as described in subsection 2.1. The system decomposition into technical components and the dependencies/interfaces between the latter are documented in the following subsections.

3.1 System Decomposition

This subsection presents a thorough list of technical components that compose the CPSoSaware system, along with short descriptions, related tasks, and partners in charge.

In comparison with D1.3, the technical components *TC2.2.3 Slice Manager* and *TC2.2.4 LightEdge* have now been removed, due to changes in the scope of the demonstrator. Specifically, there is no current need for the LightEdge component, which is a lightweight, ETSI-compliant MEC solution. Since there are no clear requirements for MEC services in the current demonstrator, a normal Edge Server solution will be used instead. Moreover, if slices are required for the purposes of the demo, these will be pre-provisioned instead of using the Slice Manager.

Furthermore, the component *TC2.1.1 Data Collection Module* described in D1.3 is no longer pertinent to the system, as it was made evident that the focus should be on the collection and analysis of factors and skills of the users as a one-time process. Therefore, the collection of user input has been performed through online questionnaires, as reported in deliverable D2.1.

Additionally, *TC4.6.1 Commissioning of Hardware Components in CPSs* has also been removed, as its functionalities have been covered by joint efforts of partners UOP and TAU in other components.

Finally, two new technical components are introduced in this version of the architecture, *TC4.5.1 Semantic Knowledge Graph* and *TC4.5.2 Semantic Knowledge Graph Service*.

Table 1 - System decomposition to Technical Components

#	ID	Name	Description	Related Tasks	Partner
1	TC2.2.1	Intra-Communication Sim Tool	Tool designed and implemented to match network requirements imposed by the application and deployed CPSoS to proposed network technologies and configurations (e.g., modulation, signal strength, duty cycle etc.) and network topologies. The tool will be based on the NS3 simulator, and it will be built based on experimentation on models of dominant wireless protocols for intra-communication, e.g., BLE, ZigBee/802.15.4, Wi-Fi.	T2.2 CPS Inter and Intra Communication Models	UOP
2	TC2.2.2	PoCL-remote	Scalable distributed OpenCL runtime layer with P2P event synchronization capabilities.	T2.2 CPS Inter and Intra Communication Models T3.2 Design and Develop CPS Layer CPSoSaware Deployment/Commissioning and Execution Mechanism.	TAU

3	TC2.3.1	Hardware Accelerator IP Cores	This refers to FPGA-based IP core components. The FPGA IP cores will be automatically generated from higher-level models by using an appropriate ML framework, whenever feasible. The IP cores will be seamlessly integrated in the PoCL-based OpenCL run-time system by means of a hardware abstraction layer (AlmaIF).	T2.3 CPS Models for HW & SW components	UOP
4	TC2.3.2	Security Accelerators for CPS security agents/sensors	FPGA based IP core components (interfaces) focused on security/cryptography.	T2.3 CPS Models for HW & SW components T2.4 Modeling of non-Functional Requirements and Security Functionality	USI
5	TC2.3.3	Model transformation to openCL	The AlmaIF protocol (interface between the host cpu and the FPGA IPs) will be appropriately extended to support the requirements of the project in terms of latency and bandwidth. Based on the extended AlmaIF protocol, specific OpenCL kernels will be enhanced by HW accelerations features in a transparent-to-the-user way.	T2.3 CPS Models for HW & SW components T3.6 Development of HW-SW Library with reliable Components T4.6 Model-based Design and Redesign of CPSoS Functional blocks Realization	UOP
6	TC2.4.1	Xilinx XRT KPI monitoring	The dynamic reconfiguration of the FPGA-based IP core components is implemented using XRT (Xilinx run-time system) functionality. Alternative HW components can be selected in real time by an application, based on environmental or other conditions. XRT allows the switching to different hardware components that can be dynamically loaded. XRT also allows the monitoring of the state of the cores and other KPIs such as the resources that have been allocated at any time.	T2.4 Modeling of non-Functional Requirements and Security Functionality T3.6 Development of HW-SW Library with reliable Components T4.1 HW-SW Partitioning Optimization based on non-Functional Requirements	UOP
7	TC2.5.1	Modeling Orchestration Tool	The modelling orchestration tool captures the CPS overall, manages individual CPS inputs and outputs between other CPSs, and orchestrates the CPSoS components in order to achieve a model of models.	T2.5 Integrated CPSoS Modeling and orchestration tools supporting autonomic functionality	8BELLS
8	TC3.1.1	Visual Localization	This component will act as a multi-modal odometer solution, aiming to fuse camera and LIDAR based SLAM. A Python library will	T3.1 CP(H)S medium: Enabling Multimodal Sensing and Embedded	ISI

			be constructed, containing state of the art SLAM algorithms, as well as the multi-modal fusion scheme. This component aims to offer increased robustness in special cases like extreme weather conditions, roads with slope, etc.	Assisted and Augment Intelligence	
9	TC3.1.2	Deep Multimodal Scene Understanding	The main objective of this module is to derive the semantic information within a given scene, namely, understanding a scene. This is the basis for autonomous driving, traffic safety, vision-guided manufacturing, or activity recognition. This module will deploy deep architectures to derive semantic information from a fusion of sensor data and the fusion of their semantic interpretation, since understanding a scene from an image or sequence of images requires more effort than simple feature extraction. RGB/Lidar, RGB/depth data will be deployed, and this module will include algorithms and deep architectures operating in distributed or centralized manner to define the operation of CPSs.	T3.1 CP(H)S medium: Enabling Multimodal Sensing and Embedded Assisted and Augment Intelligence	ISI
10	TC3.1.3	User Behaviour Monitoring	The user behavioural monitoring will be based on CPSoSaware' s collaborative sensory multi-modal fusion mechanism and will be based on algorithms for physiological and behavioural monitoring that will facilitate the evaluation of cognitive load/situational awareness development of a smart sensing module to allow inertial and optical sensor fusion, providing 6DoF pose estimation, thus dealing with occlusions and drifts. The specificities of the algorithms will be defined by the system requirements and use cases.	T3.1 CP(H)S medium: Enabling Multimodal Sensing and Embedded Assisted and Augment Intelligence	UPAT
11	TC3.1.4	AI Acceleration	The goal is the study of DCNN acceleration / compression techniques for their effective implementation in embedded platforms, lower the computational cost (number of operations, storage	T3.1 CP(H)S medium: Enabling Multimodal Sensing and Embedded Assisted and Augment Intelligence	ISI

			requirements). With the least possible loss in accuracy.		
12	TC3.2.1	PoCL-accel	This is a generic OpenCL driver (for PoCL) to interface with custom devices (hardware accelerators) from the OpenCL API.	T3.2 Design and Develop CPS Layer CPSoSaware Deployment/Commissioning and Execution Mechanism.	TAU
13	TC3.3.1	Multimodal Localization API	This component will implement a software library (written mostly in Python Programming Language) of novel techniques for multi-modal localization. Combination of LiDAR data and angle of arrival/departure will be investigated for improved cooperative localization. The studied techniques will be implemented via distributed approaches.	T3.1 CP(H)S medium: Enabling Multimodal Sensing and Embedded Assisted and Augment Intelligence T3.3 Distributed and Coalitional AI supporting autonomic intelligence	ISI
14	TC3.3.2	PathPlanning API	This component will implement a software library (written mostly in Python Programming Language) of novel techniques for collaborative path planning.	T3.1 CP(H)S medium: Enabling Multimodal Sensing and Embedded Assisted and Augment Intelligence T3.3 Distributed and Coalitional AI supporting autonomic intelligence	ISI
15	TC3.4.1	XR tools for increasing situational awareness	AR-based enhancement tools to improve the human in the loop awareness. The tools should facilitate the transfer of information (streams, reminders, or visual aids) to the user to improve focus on the current task, remember other parallel or scheduled tasks, improve response time, avoid imminent dangers or accident-related factors.	T3.4 CPHS Extended Reality based tools for increasing situational awareness	UPAT
16	TC3.5.1	CPS layer Security sensors/agents	CPS layer Security sensors/agents that collect security related data and pre-process them before transmitting them to the CPSoSaware SRMM at the system layer.	T3.5 Security and Trust Modules Design and Realization	USI
17	TC3.6.1	TCE (openasip.org) soft cores	Customized processors designed using TTA-Based Co-design Environment (TCE), an open source application-specific instruction set toolset based on the transport-triggered architecture (TTA). Various hardening features can be added	T3.6 Development of HW-SW Library with reliable Components	TAU

			via replication of functionality and special instructions.		
18	TC4.1.1	OpenCL Wrapper for Hardware IP Cores	OpenCL kernel description interface to associate Hardware IP cores with the OpenCL models.	T4.1 HW-SW Partitioning Optimization based on non-Functional Requirements	TAU
19	TC4.1.2	HW/SW profiling and analysis based on Vitis Tools	Profiling for a highly heterogeneous platform consisting of multicore ARM processor, ASIP processors as well as FPGA fixed logic IP. FPGA logic is a "morphable" computation resource without predefined computational capabilities. All SW nodes will be handled by PoCL, enabling dynamic remapping and re-scheduling opportunities.	T4.1 HW-SW Partitioning Optimization based on non-Functional Requirements	UOP
20	TC4.1.3	Architecture Optimization	This component aims to provide all necessary optimizations in order to reconfigure and redesign the System's CPSs/CPHSs so as to holistically match the systemic design and operational goals/parameters achieving reliability, robustness, responsiveness, CPS/CPHS criticality, energy efficiency, and security/trust.	T4.1 HW-SW Partitioning Optimization based on non-Functional Requirements	IBM
21	TC4.2.1	Intra-Communication Manager	Mechanisms to supervise a running network configuration in a real deployment. The metrics that reflect the application requirements will be monitored to provide feedback on whether the application requirements are met. Feedback will be extracted as a structured file by the end of each experiment on real deployments. The extracted feedback file will be used for further optimization during the simulation time. On the other hand, mechanisms allowing the reception of new network interface firmware or/and configuration file and application of these on the embedded platform.	T4.2 CPSoSaware Networking for reliable communication and cooperation between CP(H)SoS	UOP
22	TC4.3.1	Security Runtime Monitoring	This tool collects, processes, analyses and correlates security monitoring information coming from a set of heterogeneous data sources in order to detect abnormal events taking place and raise the corresponding alarms for immediate human reaction.	T4.3 CPSoSaware Security Runtime monitoring and Management (SRMM) Design and Development	ATOS

			This component operates in real-time.		
23	TC4.4.1	V2X simulator	Two simulator components: 1) Vehicle mobility simulator: Simulator based on SUMO that provide datasets with the movement of vehicles and Vulnerable Road Users in specific road networks. This simulator is able to generate collisions between vehicles and between vehicles and Vulnerable Road Users. 2) V2X message transmission simulator: Using the previous information of vehicles movement, this simulator, which is based on OMNeT++ and Vanetza, simulates the transmission of V2X messages using IEEE 802.11p and LTE-PC5 radio technologies considering realistic propagation models, taking into account the attenuation produced by buildings and the interference between messages simultaneously transmitted. The outcomes enable to obtain KPIs about the behavior of V2X communications and the improvement that applications using V2X information get. Moreover, the V2X simulator produces datasets with V2X messages received by vehicles that can be used by other CPSoSAware system components as input.	T4.4 CPSoS Simulation Tools and integration	I2CAT
24	TC4.4.3	AV Simulation	This is a simulator based on one of the available open-source solutions with additional CPSoSaware-related modules, enabling advanced simulation of all scenarios and integration with other simulations (V2X, HIL, cybersecurity, DMS, etc.). Simulation of sensors, cyberattacks, communication with vehicles and infrastructure.	T4.4 CPSoS Simulation Tools and integration	RTC
25	TC4.5.1	Semantic Knowledge Graph	W3C-compliant semantic model for representing and interrelating concepts pertinent to the two project use cases. Its aim is to store analysis results/outputs	T4.5 Cognitive System AI-assisted maintenance and CPSoS lifecycle Design Continuum Support	CTL

			from other CPSoSaware components in a homogeneous fashion.		
26	TC4.5.2	Semantic Knowledge Graph Service	Software running on-top of TC4.5.1 (Semantic Knowledge Graph) for populating the semantic model with instance data (i.e., outputs from other components) and for applying rule-based semantic reasoning for the purposes of generating alerts and reports.	T4.5 Cognitive System AI-assisted maintenance and CPSoS lifecycle Design Continuum Support	CTL
27	TC5.1.1	HLS based SW to HW Transformation	HLS based synthesized HW components with PoCL compatible interfaces.	T5.1 SW/HW Generation of non-functional property enhancements	UOP
28	TC5.3.1	Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	An AR-based CPHS user training toolkit will be developed to help the user adapt to changes in the environment and the dynamic CPSoS, whether these may concern a new machine that is added in the system or some new task process. Users often encounter strong outer constraints such as time or occupation, thus more immersive technologies aim to better exploit the uniqueness of AR and designing more effective virtual environments to improve the learning process. Virtual training scenarios will cover a broad range of user-desired activities.	T5.3 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	UPAT
29	TC5.3.2	Manufacturing Environment Simulation	This component is a simulator that provides the flexibility to simulate a workspace at various resolutions, depending on the available computational power, and different scenarios while at the same time maintaining the ability to store rich information for high-dimensional workspaces. The simulator provides immersive, gamified experiences in virtual or augmented reality during training and learning. It is also used to investigate the effect of advanced interfaces in augmented reality that increase the operators' awareness and safety in the best possible way.	T5.3 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	UPAT

Subsequently, the following subsections present technical specifications, hardware/software requirements, deployment, and interfacing details per technical component.

3.1.1 TC2.2.1 Intra-Communication Sim Tool

Type (Software/Hardware)	Software
Methodologies	Modelling of the user requirements to extract network requirements to feed the tool will be needed. The tool will initiate its mode based on heuristics and after repetitive simulations more fine - grained adjustments on the network configurations will be performed. When the simulation confirms that the application requirements are met the running network configuration will be extracted as the output of the tool and optimum operation point. When multiple network configurations meet application requirements, multiple propositions will be made, and the final configuration can be made manually or automatically based on priorities set by the application requirements (e.g. power conservation over delay minimization or vice versa etc.). When the application requirements are not met, optimum solutions will be proposed (e.g., a network configuration guaranteeing 90% mean delay and/or 60% delay jitter and/or 80% packet loss with respect to what is required).
Development Environment	C++ for model development and Python for API
Software Requirements	As introduced by NS3 simulator
Hardware Requirements	PC/VM/Cloud, Selected CPSoS embedded systems/platforms support defined intra-communication network interfaces.
Containerization	Yes
Execution Time	Depends on Simulation Scenario.
Execution Frequency	Continuously (API calls)
Main Inputs	Network configurations through HTTP requests.
Nature of Expected Input	JSON
Main Outputs	Network metrics (delay, throughput etc..)
Nature of Expected Output	JSON

3.1.2 TC2.2.2 PoCL-Remote

Type (Software/Hardware)	Software
Methodologies	Client-server and proxy software architectures are used in the layer to support efficient decentralized control of a platform with network distributed OpenCL devices.
Development Environment	C/C++/Python API available
Software Requirements	Linux. Should be easily portable to real time OSs with simple Unix compatibility layers. Even OS-less environment could be realistic if needed.
Hardware Requirements	OpenCL-supported devices. PoCL can be used to provide OpenCL support for various devices.
Containerization	Yes
Execution Time	Application specific
Execution Frequency	Application specific
Main Inputs	OpenCL application definitions along with their kernels. OpenCL driver for proprietary GPUs and custom devices (via PoCL).
Nature of Expected Input	C/C++/Python programs, preferably split to OpenCL host and kernel programs.

Main Outputs	Application specific
Nature of Expected Output	ASCII or raw files, object detection labels, or what applies to the case at hand.

3.1.3 TC2.3.1 HW Accelerator IP Cores

Type (Software/Hardware)	Hardware and associated Software drivers
Methodologies	Hardware component design and implementation techniques (VHDL based design, efficiency optimization techniques)
Development Environment	The component will be developed using hardware description language (e.g VHDL) and possible association with OpenCL.
Software Requirements	Embedded system and hardware design and development tools for IP Core creation.
Hardware Requirements	Embedded devices with System on Chip that has FPG fabric (e.g., Xilinx Zynq, Ultrascale).
Containerization	N/A
Execution Time	Within milliseconds.
Execution Frequency	Operating as part of a HW/SW partitioning environment whenever hardware acceleration is needed.
Main Inputs	Specifications on ML algorithms.
Nature of Expected Input	Formal specifications, Model of computing components
Main Outputs	Hardware IP Cores for Acceleration of ML operation
Nature of Expected Output	Model in VHDL, Hardware IP core library components

3.1.4 TC2.3.2 Security Accelerators for CPS security agents/sensors

Type (Software/Hardware)	Hardware and associated Software drivers
Methodologies	Hardware component design and implementation techniques (VHDL based design, efficiency optimization techniques, security strengthening techniques for security components).
Development Environment	The component will be developed using hardware description language (e.g VHDL) and possible association with OpenCL.
Software Requirements	Embedded system and hardware design and development tools for IP Core creation.
Hardware Requirements	Embedded devices with System on Chip that has FPGA fabric (e.g., Xilinx Zynq, Ultrascale).
Containerization	N/A
Execution Time	Within milliseconds.
Execution Frequency	Operating as part of a HW/SW partitioning environment whenever hardware acceleration is need.
Main Inputs	Specifications on Security approach and CPS security sensors/agents.
Nature of Expected Input	Formal specifications, Model of computing components.
Main Outputs	Hardware IP Cores for Acceleration of Security/Cryptography.
Nature of Expected Output	Model in VHDL, Hardware IP core library components.

3.1.5 TC2.3.3 Model transformation to OpenCL

Type (Software/Hardware)	Hardware and associated Software drivers
Methodologies	The AlmaIF protocol (interface between the host CPU and the FPGA IPs) will be appropriately extended to support the requirements of the project in terms of latency and bandwidth. Based on the extended AlmaIF protocol, specific OpenCL kernels will be enhanced by HW accelerations features in a transparent-to-the-user way.
Development Environment	Xilinx Vitis, HLS, Vivado, C++, OpenCL
Software Requirements	Embedded system and hardware design and development tools for IP Core creation.
Hardware Requirements	Embedded devices with System on Chip that has FPGA fabric (e.g., Xilinx Zynq, Ultrascale).
Containerization	N/A
Execution Time	Within milliseconds
Execution Frequency	Application specific
Main Inputs	OpenCL kernel programs and high-performance FPGA buses specifications
Nature of Expected Input	OpenCL kernels and C/C++ python programs
Main Outputs	Portable and plug-and-play FPGA IP cores
Nature of Expected Output	Hardware IP cores (in HLS) and associated Linux device drivers

3.1.6 TC2.4.1 Xilinx XRT KPI monitoring

Type (Software/Hardware)	Software
Methodologies	Dynamic loading of hardware core bitsreams stored locally or remotely, monitoring the status of the employed hardware cores.
Development Environment	Xilinx Vitis, XRT, Vivado, C++, OpenCL VHDL
Software Requirements	host computer with Ubuntu OS where Xilinx Vitis, XRT and Vivado have been installed. Petalinux OS runs on the target FPGA system.
Hardware Requirements	At least 16GB RAM, 200GB of storage and i5 (or faster architecture) is required for the host computer where the applications are developed. Target boards require Ultrascale architectures.
Containerization	Yes. XRT offers a container environment where the applications are developed based on specific templates.
Execution Time	In the order of few milliseconds.
Execution Frequency	Target: 200MHz or higher.
Main Inputs	For the DSM application the inputs can be the environmental lighting conditions, gender information, etc.
Nature of Expected Input	Environmental sensors or outputs of other modules (e.g., face recognition).
Main Outputs	List of hardware cores running, their speed, memory requirements, resources allocated, etc.
Nature of Expected Output	KPIs represented as numbers, sizes, descriptions etc.

3.1.7 TC2.5.1 Modelling Orchestration Tool

Type (Software/Hardware)	Software
Methodologies	Agile Software Development
Development Environment	Python
Software Requirements	UI libraries in Python (Tkinder) and Jenkins pipelines for CI/CD.
Hardware Requirements	Standard Desktop
Containerization	No
Execution Time	Within seconds
Execution Frequency	Event-driven
Main Inputs	Selection of Simulation tools and configurations from user.
Nature of Expected Input	Selection from user via UI.
Main Outputs	Decisions on how each CPS simulation model will be orchestrated at event, KPIs to drive the decision and model adjustment.
Nature of Expected Output	Results for each simulator individually.

3.1.8 TC3.1.1 Visual Localization

Type (Software/Hardware)	Software
Methodologies	LiDAR and image processing: ring estimation, curb detection, road estimation using the candidate curbs, object recognition using classification approaches.
Development Environment	Python
Software Requirements	Vision problems via multi-modal odometers solutions.
Hardware Requirements	Embedded devices with AI accelerators (e.g., Xilinx Vitis).
Containerization	No
Execution Time	Within milliseconds.
Execution Frequency	In each incoming scene/frame.
Main Inputs	LiDAR Data, Images from PASEU, Simulator (e.g., CARLA) and available datasets (e.g., Lyft).
Nature of Expected Input	Data from CAN bus, image Files, point cloud files.
Main Outputs	Location estimation of vehicle, location estimation of obstacles, robustification to data anomalies and outliers.
Nature of Expected Output	Time series

3.1.9 TC3.1.2 Deep Multimodal Scene Understanding

Type (Software/Hardware)	Software
Methodologies	Vision problems via deep learning solutions.
Development Environment	Python, TensorFlow
Software Requirements	TensorFlow API

Hardware Requirements	Embedded devices with AI accelerators (e.g., Xilinx Vitis).
Containerization	Yes
Execution Time	40 fps
Execution Frequency	In each incoming scene/frame.
Main Inputs	The dataset used for training will be based on 3 major pillars: (a) synthetic dataset, (b) real dataset, (c) augmented data. Throughout the following subsections the criteria for structuring the dataset will be extensively discussed.
Nature of Expected Input	Images, point clouds
Main Outputs	Score and detected objects in scenes.
Nature of Expected Output	Annotated images and point clouds.

3.1.10 TC3.1.3 User Behaviour Monitoring

Type (Software/Hardware)	Software
Methodologies	Development of vision technology based on onboard units will provide measures of drowsiness. Direct measures, such as eye closure and blink analysis. Indirect measures based on longitudinal and lateral control.
Development Environment	Python, C++, MATLAB
Software Requirements	No special software requirements.
Hardware Requirements	Video capturing devices.
Containerization	Yes
Execution Time	Real-time
Execution Frequency	Continuously (event-based)
Main Inputs	Direct measures of drowsiness: eye closure and blink analysis Indirect measures of drowsiness: Driving performance measure: speed (mean and variability) and distance control (distance to lead vehicle). Driver activity measure: pedal activity (driver's use of pedals). Driving performance measure: lane keeping performance (standard deviation of the lateral position, time to line crossing, number of lane crossings, mean lateral position, mean yaw rate). Driver activity measure: steering behaviour (magnitude and frequency of steering activity), steering variability, slow and fast steering corrections.
Nature of Expected Input	Data from CAN bus, Images Files, real time data streams, video.
Main Outputs	Level of Drowsiness (Indicator values between a range of measurements), warning signs.
Nature of Expected Output	Time series, text

3.1.11 TC3.1.4 AI Acceleration

Type (Software/Hardware)	Software
Methodologies	We focus on the following methodologies for model compression and acceleration:

	1) Parameter pruning: Unimportant parameters (e.g., filters) are removed and not considered during the inference phase of the DCNN deployment. 2) Codebook-based parameter sharing: Such approaches aim at increasing common representations of the involved parameters via the design of codebooks (e.g., k-means-based, dictionary-learning-based).
Development Environment	Python, MATLAB
Software Requirements	Machine learning libraries in Python / MATLAB. Auxiliary libraries like for input/output functions, programming interface and evaluation.
Hardware Requirements	Embedded devices with AI accelerators (e.g., Xilinx Vitis).
Containerization	No
Execution Time	Within milliseconds.
Execution Frequency	Between 15 and 30 fps.
Main Inputs	LiDAR data, images from PASEU, simulator (e.g., CARLA) and available datasets (e.g., Lyft).
Nature of Expected Input	Data from CAN bus, images files, point cloud files.
Main Outputs	Accelerated networks in ONNX.
Nature of Expected Output	Time series

3.1.12 TC3.2.1 PoCL-Accel

Type (Software/Hardware)	Software
Methodologies	The method to provide portability across FPGA-based IPs and other hardware accelerators, programmable and non-programmable is done through standardized IP wrappers which are interfaced with a common software driver integrated to the PoCL OpenCL driver framework.
Development Environment	C/C++, Linux
Software Requirements	C compiler
Hardware Requirements	FPGA chip. SoC and discrete PCIe card-based ones have been tested.
Containerization	Yes
Execution Time	Depends on the accelerated function.
Execution Frequency	Depends on the accelerated function.
Main Inputs	Generic, depends on the accelerator.
Nature of Expected Input	Generic, depends on the accelerator.
Main Outputs	Generic, depends on the accelerator.
Nature of Expected Output	Generic, depends on the accelerator.

3.1.13 TC3.3.1 Multimodal Localization API

Type (Software/Hardware)	Software
Methodologies	Cooperative localization (CL) has been an active area of research over the last years as it presents several advantages over traditional single vehicle, robots and mobile IoT systems, like robustness, increased efficiency, information exchange and increased virtual sensing capability. It has received extensive interest from robotics,

	optimization and wireless communication communities. Due to the aforementioned benefits, CL has been widely used in a large variety of applications, including navigation of autonomous ground and aerial vehicles, target tracking and time synchronised path following.
Development Environment	Python
Software Requirements	Machine learning libraries in Python. Auxiliary libraries like for input/output functions, programming interface and evaluation.
Hardware Requirements	Embedded devices with AI accelerators (e.g., Xilinx Vitis).
Containerization	Yes
Execution Time	Within milliseconds, depending on the hardware and network capabilities.
Execution Frequency	Every 100-300 ms (depending on the arrival of measurements from GPS and neighbors).
Main Inputs	Simulator (e.g., CARLA) and available datasets.
Nature of Expected Input	JSON, data from CAN bus.
Main Outputs	Collaborative algorithms that accurately estimate nearby objects' locations.
Nature of Expected Output	Python script files.

3.1.14 TC3.3.2 PathPlanning API

Type (Software/Hardware)	Software
Methodologies	A mobile network system is deployed in a cluttered environment to achieve common mission within a changing environment. This networked vehicle system experiences some uncertainties in navigation and sensing during its mission. On the other hand, networked vehicles must plan their paths to avoid collision with environmental obstacles and other vehicles. Moreover, vehicles should maintain certain connectivity constraints such as the coordination task can be accomplished. Since vehicles are to provide a wireless communication infrastructure, the motion planning problem needs to incorporate wireless communication constraints. Moreover, Human-in-the-loop constraints will also have to be incorporated, to include indirect measures of drowsiness. This will create a multi-objective decision-making problem in which optimum motion planning decisions considering only one criterion may not be the best solution to cover all objectives and limitations.
Development Environment	Python
Software Requirements	Machine learning libraries in Pyhton. Auxiliary libraries like for input/output functions, programming interface and evaluation.
Hardware Requirements	Embedded devices with AI accelarators (e.g., Xilinx Vitis).
Containerization	Yes
Execution Time	Within milliseconds, depending on the hardware and network capabilities.
Execution Frequency	Every 100-300 ms (depending on the arrival of measurements from GPS and neighbors)
Main Inputs	Simulator (e.g., CARLA) and available datasets.
Nature of Expected Input	JSON, data from CAN bus.
Main Outputs	Collaborative algorithms that optimally determine the future driving actions of self and nearby vehicles.
Nature of Expected Output	Python script files.

3.1.15 TC3.4.1 XR tools for increasing situational awareness

Type (Software/Hardware)	Software
Methodologies	Different augmented reality techniques using AR Glasses, mobile devices, and marker-less tracking, will be implemented. Multi-modal projections that visually describe the significant environment parameters will act as guidelines through situations with high task load in challenging and multi-tasking environments.
Development Environment	Python, C++, MATLAB
Software Requirements	No special software requirements.
Hardware Requirements	Devices that could provide AR output.
Containerization	Yes
Execution Time	Within milliseconds or in the range of a LIDAR frame.
Execution Frequency	Continuously (event-based).
Main Inputs	Eye gaze / head direction / hand gestures. Physiological values of users (temperature, heart rate, inhalation, etc.,). Environmental values, measurements by sensors and feedback by other parallel processes of the pipeline. Captured scene (e.g., point cloud) for an objects-of-interest analysis or scene analysis and understanding. Localization (GPS). Personalized information related to each user. Collaborative information (e.g., from other drivers or colleagues).
Nature of Expected Input	Three types: 1) real-time data streams (time series) and 2) historic measurements (scalars or time series) 3) 3D points clouds or meshes (geometry).
Main Outputs	Pop up messages (alerts, warnings, notificasions/suggestions, information). Real time visualization (i.e., transparent plans, trajectories etc.). Augmented static or dynamic 3D objects and simulation. Sounds.
Nature of Expected Output	Text, time series, 3D geometry, sound.

3.1.16 TC3.5.1 CPS layer Security sensors/agents

Type (Software/Hardware)	Hardware and associated Software drivers
Methodologies	Low-level C code driver development and trusted OS integration using dedicated security/cryptography hardware accelerator peripherals.
Development Environment	Xilinx Software Development Kit
Software Requirements	Embedded system and hardware design and development tools for IP Core creation.
Hardware Requirements	Embedded devices with System on Chip that have FPGA fabric (e.g., Xilinx Zynq, Ultrascale).
Containerization	N/A
Execution Time	Within milliseconds.
Execution Frequency	Operating as part of a HW/SW partitioning environment whenever security functionality if needed (running constantly on the CPS layer)

Main Inputs	Specifications on Security approach and CPS security sensors/agents, Hardware dedicated IP Core accelerator models and libraries.
Nature of Expected Input	Formal specifications, IP Core libraries (also posibly OpenCL kernels).
Main Outputs	Executables and security-hardened/trusted OS structures, kernel mode security drivers.
Nature of Expected Output	Executable software

3.1.17 TC3.6.1 TCE (openasip.org) soft cores

Type (Software/Hardware)	Hardware and Software
Methodologies	TCE is an application-specific processor design and programming toolset. Its open-source repository is also called OpenASIP due to it being developed to support a wider set of programming models than the original Transport Triggered Architecture, which is in the core of the processor template of the toolset. The tool supports a wide range of customization options, including register files, function units and their operations, datapath connectivity, with retargetable instruction-set simulators, compilers and RTL generators driven by an architecture description file.
Development Environment	C/C++/VHDL
Software Requirements	Tools run on Linux-based desktops, possibly MacOS (unsupported).
Hardware Requirements	Application specific.
Containerization	Yes
Execution Time	Application specific.
Execution Frequency	Application specific.
Main Inputs	C/C++/OpenCL C description of the application and the non-functional requirements.
Nature of Expected Input	Preferably a C/C++/OpenCL C implementation which can be gradually transferred (co-designed) to a customized processor.
Main Outputs	Application specific.
Nature of Expected Output	ASCII or raw files, object detection labels, or what applies to the case at hand.

3.1.18 TC4.1.1 OpenCL Wrapper for Hardware IP Cores

Type (Software/Hardware)	Hardware and associated Software drivers
Methodologies	OpenCL kernel specification
Development Environment	Xilinx toolbox (Vitis, SDx Tools) or similar opensource components, PoCL.
Software Requirements	Embedded system and hardware design and development tools for IP Core creation.
Hardware Requirements	Embedded devices with System on Chip that has FPG fabric (e.g., Xilinx Zynq, Ultrascale).
Containerization	N/A
Execution Time	Within milliseconds.
Execution Frequency	Operating as part of a HW/SW partitioning environment whenever hardware acceleration is need.
Main Inputs	Hardware IP Core models and IP Core libraries.

Nature of Expected Input	IP Core libraries, specification, models, interfaces.
Main Outputs	OpenCL kernels for integration to OpenCL schemas and CPS models.
Nature of Expected Output	OpenCL kernel libraries.

3.1.19 TC4.1.2 HW/SW profiling and analysis based on Vitis Tools

Type (Software/Hardware)	Software
Methodologies	Profiling is used to recognize the computationally intensive modules that can be accelerated in hardware in a heterogeneous platform consisting of multicore ARM processor, ASIP processors as well as reconfigurable hardware. Xilinx Vitis will be used to offer accurate profiling and speed estimation mechanisms.
Development Environment	Xilinx Vitis, XRT, Vivado, C++, OpenCL
Software Requirements	Ubuntu OS on the host computer where Xilinx Vitis, XRT and Vivado have been installed. The OS running on the target FPGA system is Petalinux.
Hardware Requirements	At least 16GB RAM, 200GB of storage and i5 (or faster architecture) is required for the host computer where the applications are developed. Target boards with Ultrascale architecture are used.
Containerization	Yes, Vitis offers a containerized environment where profiling can be performed.
Execution Time	Profile time of the components analysed are expected to range up to some hundreds of milliseconds.
Execution Frequency	Target: 200MHz or higher
Main Inputs	Simulation stimuli, application to profile.
Nature of Expected Input	List of events that occur in the input signals, platform under analysis with the implementation of the components (C++ or VHDL).
Main Outputs	Latency of the various components (hardware and software).
Nature of Expected Output	Latency measured in milliseconds. Average values extracted statistically.

3.1.20 TC4.1.3 Architecture Optimization

Type (Software/Hardware)	Software
Methodologies	Mathematical optimization
Development Environment	Python, PyCharm, C++
Software Requirements	IBM CPLEX or another MILP solver
Hardware Requirements	16GB RAM, 100GB storage, i7 or faster CPU
Containerization	Docker, Kubernetes
Execution Time	90s
Execution Frequency	Design time
Main Inputs	Hardware platforms, desired functionality
Nature of Expected Input	List of hardware platforms and software libraries together with capabilities, list of functional units with requirements and inter-dependencies
Main Outputs	Execution locations

Nature of Expected Output	List of functional units together with hardware platform or software library where the functionality will be implemented

3.1.21 TC4.2.1 Intra-Communication Manager

Type (Software/Hardware)	Software
Methodologies	Open network stacks will be used to implement watchers, watchdogs and monitoring structures based on application requirements and network performance metrics.
Development Environment	Python Development Environment
Software Requirements	Open network stacks
Hardware Requirements	At least 2 network interfaces available supporting 2 different selected wireless technologies for intra - communication accordingly.
Containerization	Yes
Execution Time	Depends on the underlying network interface.
Execution Frequency	Continuously (event-based)
Main Inputs	Intra layer communication requirements through MQTT messages
Nature of Expected Input	JSON
Main Outputs	Available Open network stacks configurations to meet the input requirements. Assessment of network performance with regards to application requirements. Report of the configuration applied to the target platform.
Nature of Expected Output	Network configuration files.

3.1.22 TC4.3.1 Security Runtime Monitoring

Type (Software/Hardware)	Software
Methodologies	The development of the SRMM does not follow any specific methodology. Very elementary cohabitation rules have been applied in the internal GitLab environment leveraged for implementation activities, which have shown to suffice for the achievement of the different milestones.
Development Environment	An internal Gitlab environment has been setup. The different extensions are developed in separate branches and smoothly integrated in the main one performing rigorous testing sessions, including regression tests.
Software Requirements	A Linux OS has to be in place to run the SRMM. If the dockerized version will be run, it must include Docker in the installation.
Hardware Requirements	Between 30 and 50 Gbs of free disk space. For processing capabilities, a minimum of 4 virtual cores is requested, being convenient to have 8 available. As for RAM memory, 4Gb could suffice for very optimized testing sessions, but 8 Gb are much more convenient.
Containerization	Yes
Execution Time	Around 500 ms from event collection to alarm generation.
Execution Frequency	Constantly receiving events.
Main Inputs	Events and logs received, through syslog, from tools and security probes.

Nature of Expected Input	JSON with security alert information.
Main Outputs	Information about an incident detected (affected device, type of incident, etc).
Nature of Expected Output	Security alerts as a JSON.

3.1.23 TC4.4.1 V2X simulator

Type (Software/Hardware)	Software
Methodologies	The simulator is implemented in two blocks: 1) Vehicle mobility generator: This block is based in SUMO which has been upgraded to build scenarios of specific road networks with rear-end collisions, collisions produced because of a red-light violation and the inclusion of Vulnerable Road Users (pedestrians and bicycles) with predetermined trajectories in sidewalks, in bicycle's lanes and in standard vehicle's lanes. This mobility generator will be used to build up the scenarios of task T6.3 during the third year of the project. 2) V2X message transmission: This simulator, which is based in OMNeT++ and Vanetza, takes the road network map and the vehicle mobility pattern, previously generated by the vehicle mobility generator, and simulates the V2X message transmission between vehicles using IEEE 802.11p and LTE-PC5 radio technologies plus ETSI G5 communication protocol stack. The raw output of this simulation is the time and position where vehicles have transmitted and received V2X messages. With this information, another software tool computes the transmission packet error ratio, neighbour awareness ratio, errors between the registered position of specific vehicles in the Local Dynamic Map and their real position and other statistics that may be implemented during the third year of CPSoSAware project. Additionally, the traces of V2X messages received by specific vehicles can be used as input of other components of the CPSoSAware system. The main interest of this data is that it has been computed with high precision transmission models taking into account radio propagation models, presence of buildings which attenuate the radio signal, interferences between simultaneous transmissions of different vehicles and the simulation of the Medium Access Control mechanism used for the radio technology.
Development Environment	Simulator is developed in C++ and phyton.
Software Requirements	OMNeT++, SUMO, Vanetza
Hardware Requirements	Standard Intel PC (e.g. i7).
Containerization	Yes
Execution Time	Largely dependent on the complexity of the model (e.g. number of cars). A complex simulation can take > 24 hours.
Execution Frequency	N/A
Main Inputs	Configuration parameters of the use case to study: • Geographical situation: Road network (lanes, intersections, traffic signals, ...). • Involved actors: Number of vehicles, presence of Vulnerable Road Users (pedestrians and bicycles). • Actor's behaviour: Vehicle's and Vulnerable Road Users' trajectories, presence, and type of collisions between vehicles and/or Vulnerable Road Users. • V2X message transmission: Type of messages, size of payload, propagation model.

Nature of Expected Input	Configuration files which some of them are provided by OMNeT++ and SUMO frameworks and others are built using specific applications. They require specific information depending on the simulated use case: • Road network is manually defined or obtained using data of OpenStreetMap. • Number and trajectories of vehicles and Vulnerable Road Users are defined manually or in an autonomous random way. Vehicles on the same lane use the Krauss car-following model. • Collisions are produced depending on the driver's behaviour that is set manually. • V2X communication parameters are set manually.
Main Outputs	1) Statistics about the performance parameters of: • Influence of V2X communications in the accuracy of the knowledge that vehicles may have about the real position of their neighbour vehicles and Vulnerable Road Users. • Time to collision computation. • Dangerous level of driving events. • V2X communication protocol and radio technology. 2) File with the V2X messages that vehicles receive. This information is used as input of other components of CPSoSAware system.
Nature of Expected Output	Any type of text file (e.g. CSV).

The following table specifies the V2X Simulator solution developed by ROBOTEC.

Type (Software/Hardware)	Software
Methodologies	Mathematical modeling of wave propagation, separation of layers and modular structure – V2X simulator can be run on different computer than simulator
Development Environment	Simulator is developed in C++
Software Requirements	Depends on selected open-source simulator (Carla, LGSVL simulator, AirSim, Robotec Simulator), needs ROS2.
Hardware Requirements	Standard Intel PC (e.g. i7).
Containerization	Yes
Execution Time	Dependent on the complexity of the model (e.g. number of cars, propagation model).
Execution Frequency	N/A
Main Inputs	Environment, vehicles position, dynamic objects, propagation model
Nature of Expected Input	ROS2 messages with simple structures
Main Outputs	Distance beetween vehicles
Nature of Expected Output	ROS2 message with distance beetween certain vehicle and all others

3.1.24 TC4.4.3 AV Simulation

Type (Software/Hardware)	Software
Methodologies	Agile, modular structure

Development Environment	Depends on selected open-source simulator. After state-of-the-art analysis of available simulators performed in D1.1, the following simulators were selected: Carla, LGSVL simulator, Robotec Simulator.
Software Requirements	Depends on selected open-source simulator (Carla, LGSVL simulator, AirSim, Robotec Simulator).
Hardware Requirements	Depends on selected open-source simulator (Carla, LGSVL simulator, AirSim, Robotec Simulator).
Containerization	Yes
Execution Time	Depends on scenario.
Execution Frequency	> 20FPS
Main Inputs	Vehicle models, control algorithms, predefined control scenarios.
Nature of Expected Input	Configuration files defining test scenarios
Main Outputs	Collected Data, metrics for validation, photorealistic visualization of system behaviour.
Nature of Expected Output	Datasets for perception (images, point clouds and CSV with labels), reports of scenarios validation.

3.1.25 TC4.5.1 Semantic Knowledge Graph

Type (Software/Hardware)	Software
Methodologies	NeOn ontology engineering methodology; adoption of W3C-recommended standards.
Development Environment	Protégé ontology editor
Software Requirements	SPARQL-enabled RDF triplestore (e.g., GraphDB free edition)
Hardware Requirements	8GB RAM, 100 GB SSD
Containerization	Yes
Execution Time	N/A
Execution Frequency	Continuous
Main Inputs	Analysis results generated by other CPSoSaware components
Nature of Expected Input	Parameterizable SPARQL queries
Main Outputs	Results from execution of rules: e.g., risk levels, error estimations etc.
Nature of Expected Output	SPARQL query result-sets as JSON.

3.1.26 TC4.5.2 Semantic Knowledge Graph Service

Type (Software/Hardware)	Software
Methodologies	Semantic data integration (ontology population); rule-based semantic reasoning.
Development Environment	Python
Software Requirements	REST API libraries, SPARQLWrapper, TC4.5.1 Semantic Knowledge Graph
Hardware Requirements	2GB RAM, 1GB SSD
Containerization	Yes

Execution Time	Varies depending on the complexity of requests. Usually within 1-3 seconds.
Execution Frequency	Continuously (event-based).
Main Inputs	Post and Get HTTP requests from other CPSoSaware components for the insertion of knowledge to and retrieval of knowledge from TC4.5.1 correspondingly.
Nature of Expected Input	JSON
Main Outputs	Knowledge stored in TC4.5.1, augmented with results from semantic reasoning.
Nature of Expected Output	JSON

3.1.27 TC5.1.1 HLS based SW to HW Transformation

Type (Software/Hardware)	Software
Methodologies	HLS based synthesized HW components with PoCL compatible interfaces.
Development Environment	Xilinx Vitis HLS, Vivado HLS, C/C++/VHDL, OpenCL
Software Requirements	Ubuntu OS on the host computer where Xilinx Vitis, XRT and Vivado have been installed. The OS running on the target FPGA system is Petalinux.
Hardware Requirements	At least 16GB RAM, 200GB of storage and i5 (or faster architecture) is required for the host computer where the applications are developed. Target boards with Ultrascale or Zynq architecture are used.
Containerization	Yes, Vitis offers a containerized environment for several services.
Execution Time	Up to several minutes.
Execution Frequency	Several times During the development of an application.
Main Inputs	Software modules
Nature of Expected Input	Descriptions in C++/OpenCL
Main Outputs	Bitstreams of the hardware kernels
Nature of Expected Output	HLS based synthesized HW components with PoCL compatible interfaces.

3.1.28 TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS

Type (Software/Hardware)	Software
Methodologies	Different augmented reality techniques using AR Glasses, mobile devices and marker-less tracking, will be implemented. The task will develop off-site training solutions.
Development Environment	Python, C++, MATLAB
Software Requirements	No special software requirements
Hardware Requirements	Devices that could provide AR output.
Containerization	Yes
Execution Time	Within milliseconds.
Execution Frequency	Continuously (event-based)
Main Inputs	Eye gaze, head direction, hand gestures. Physiological values of the user (temperature, heart rate, inhalation, etc.,). Environmental values, measurements by sensors and feedback by other parallel processes of the pipeline.

	Captured scene (e.g., point cloud). Personalized information related to each user. Collaborative information (e.g., from the instructor or other colleagues).
Nature of Expected Input	Three types: 1) real-time data streams (time series) and 2) historic measurements (scalars or time series) 3) 3D points clouds or meshes (geometry).
Main Outputs	XR-based toolkit for training in VR, warning signs for the safety of human operator or driver, 3D visualization of safe zones in AR/VR, a driving simulator with signs and visual hints.
Nature of Expected Output	Text, time series, 3D geometry and graphics.

3.1.29 TC5.3.2 Manufacturing Environment Simulation

Type (Software/Hardware)	Software
Methodologies	Emulation of robotic movements in a digital workspace in Unity3D as a prior knowledge during runtime computation of (spatiotemporal) human-robot collision risk. Different rendering methods in Unity in VR/AR for static/dynamic safety zones
Development Environment	C#, C++, Unity3D
Software Requirements	No special software requirements
Hardware Requirements	Devices that could provide VR/AR output and/or display monitor.
Containerization	Yes
Execution Time	Within milliseconds.
Execution Frequency	>= 30FPS
Main Inputs	Human behaviour models, predefined control scenarios, robot's trajectory, operator's anthropometrics.
Nature of Expected Input	Robot models, predefined scenarios
Main Outputs	Collected Data, metrics for validation, photorealistic visualization of system behaviour.
Nature of Expected Output	Datasets for perception (images and CSV with labels). Reports of scenarios validation.

3.2 Dependencies and Interfaces Between System Components

Table 2 collectively presents the dependencies between technical components, as defined by the technical partners. A dependency may be defined as a fulfilled or scheduled interaction of a component with another, where there is usually an exchange of information, a trigger, or an integrated functionality.

Table 2 - Dependencies between Technical Components

Technical component interacts with / depends on	TC2.2.1 Intra-Communication Sim Tool	TC2.2.2 PoCL-remote	TC2.3.1 ML Hardware Accelerator IP Cores	TC2.3.2 Security Accelerators for CPS security agents/ sensors	TC2.3.3 Model transformation to openCL	TC2.4.1 Xilinx XRT KPI monitoring	TC2.5.1 Modelling Orchestration Tool	TC3.1.1 Visual Localization	TC3.1.2 Deep Multimodal Scene Understanding	TC3.1.3 User Behaviour Monitoring	TC3.1.4 AI Acceleration	TC3.2.1 PoCL-accel	TC3.3.1 Multimodal Localization API	TC3.3.2 PathPlanning API	TC3.4.1 XR tools for increasing situational awareness	TC3.5.1 CPS layer Security sensors/agents	TC3.6.1 TCE (openasip.org) soft cores	TC4.1.1 OpenCL Wrapper for Hardware IP Cores	TC4.1.2 HW/SW profiling and analysis based on Vitis Tools	TC4.1.3 Architecture Optimization	TC4.2.1 Intra-Communication Manager	TC4.3.1 Security Runtime Monitoring	TC4.4.1 V2X simulator	TC5.3.2 Manufacturing Environment Simulation	TC4.4.3 AV Simulation	TC4.5.1 Semantic Knowledge Graph	TC4.5.2 Semantic Knowledge Graph Service	TC5.1.1 HLS based SW to HW Transformation	TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS
TC2.2.1 Intra-Communication Sim Tool	✗	✓					✓																						
TC2.2.2 PoCL-remote	✓	✗			✓						✓	✓	✓				✓	✓											
TC2.3.1 ML Hardware Accelerator IP Cores			✗						✓			✓																	
TC2.3.2 Security Accelerators for CPS security agents/sensors				✗																									
TC2.3.3 Model transformation to openCL		✓			✗				✓																				
TC2.4.1 Xilinx XRT KPI monitoring						✗						✓																	
TC2.5.1 Modelling Orchestration Tool	✓						✗														✓		✓	✓	✓				✓
TC3.1.1 Visual Localization								✗	✓				✓																
TC3.1.2 Deep Multimodal Scene Understanding			✓		✓			✓	✗		✓	✓	✓	✓					✓	✓							✓		
TC3.1.3 User Behaviour Monitoring										✗					✓												✓		
TC3.1.4 AI Acceleration									✓		✗																		
TC3.2.1 PoCL-accel		✓	✓			✓			✓			✗					✓	✓											

TC3.3.1 Multimodal Localization API		✓					✓	✓				✗	✓								✓	✓						
TC3.3.2 PathPlanning API								✓				✓	✗								✓	✓						
TC3.4.1 XR tools for increasing situational awareness									✓					✗														
TC3.5.1 CPS layer Security sensors/agents															✗					✓	✓							
TC3.6.1 TCE (openasip.org) soft cores		✓									✓					✗	✓											
TC4.1.1 OpenCL Wrapper for Hardware IP Cores		✓									✓					✓	✗											
TC4.1.2 HW/SW profiling and analysis based on Vitis Tools								✓										✗										
TC4.1.3 Architecture Optimization								✓											✗									
TC4.2.1 Intra-Communication Manager						✓														✗								
TC4.3.1 Security Runtime Monitoring													✓		✓						✗							
TC4.4.1 V2X simulator						✓							✓		✓							✗						
TC5.3.2 Manufacturing Environment Simulation						✓																	✗					
TC4.4.3 AV Simulation						✓																		✗				
TC4.5.1 Semantic Knowledge Graph																									✗	✓		
TC4.5.2 Semantic Knowledge Graph Service								✓	✓																✓	✗		
TC5.1.1 HLS based SW to HW Transformation																											✗	
TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS						✓																						✗

For the representation of interfaces, we created UML diagrams using the open-source *PlantUML*[2] language and the online *PlantText UML editor*[3], that allow the generation of diagrams from simple textual descriptions. Figures 2 to 6 offer a closer look to the designed interfaces of specific component clusters, while Figure 7 presents an overview of all system interfaces. The respective PlantUML code is provided in Annex A.

[2] https://plantuml.com/

[3] https://planttext.com/

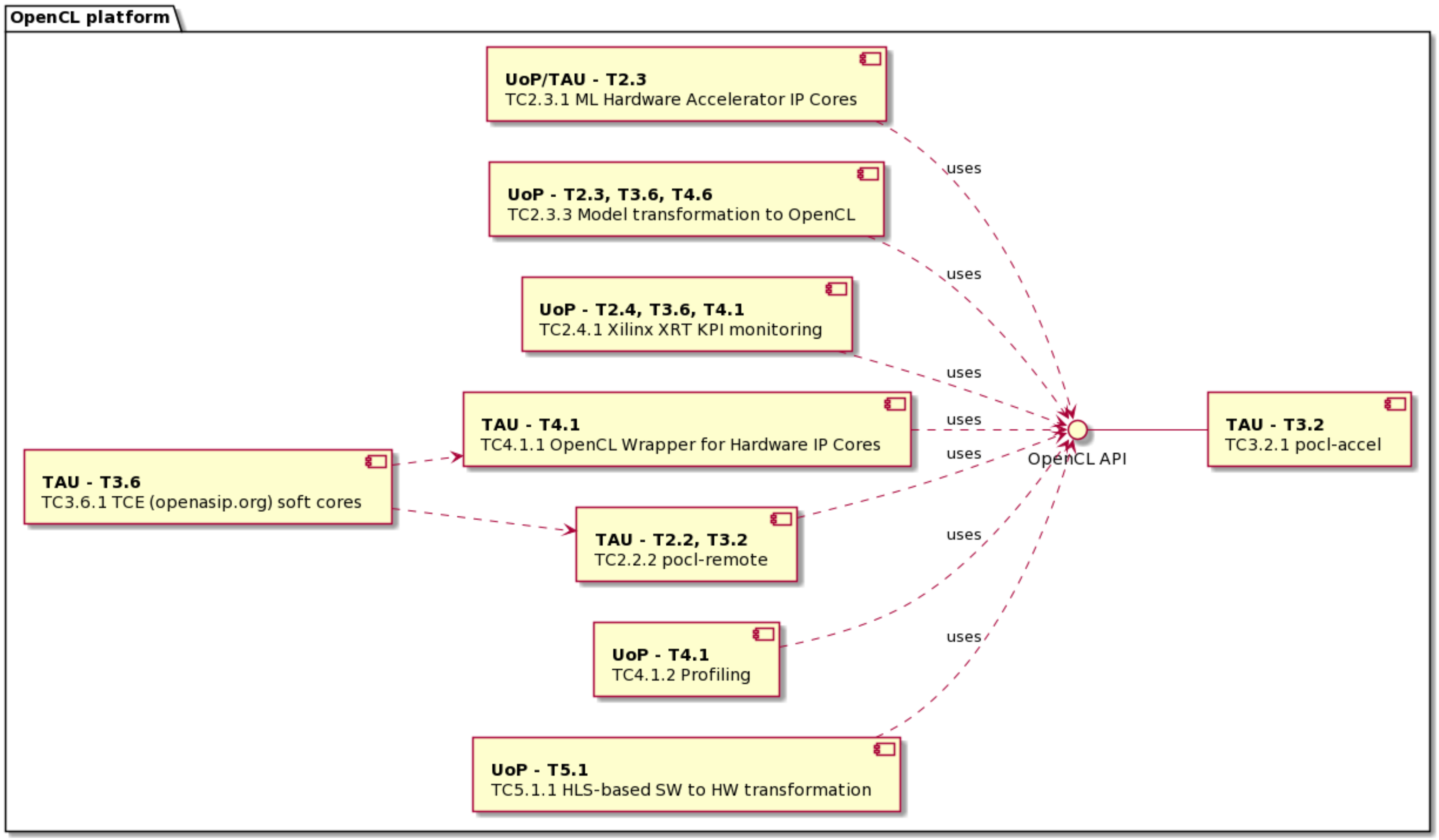
OpenCL platform
UoP/TAU - T2.3
TC2.3.1 ML Hardware Accelerator IP Cores
UoP - T2.3, T3.6, T4.6
TC2.3.3 Model transformation to OpenCL
UoP - T2.4, T3.6, T4.1
TC2.4.1 Xilinx XRT KPI monitoring
TAU - T4.1
TC4.1.1 OpenCL Wrapper for Hardware IP Cores
TAU - T3.6
TC3.6.1 TCE (openasip.org) soft cores
TAU - T2.2, T3.2
TC2.2.2 pocl-remote
UoP - T4.1
TC4.1.2 Profiling
UoP - T5.1
TC5.1.1 HLS-based SW to HW transformation
uses
uses
uses
uses
uses
uses
uses
OpenCL API
TAU - T3.2
TC3.2.1 pocl-accel

Figure 2 - Interfaces of components related to OpenCL

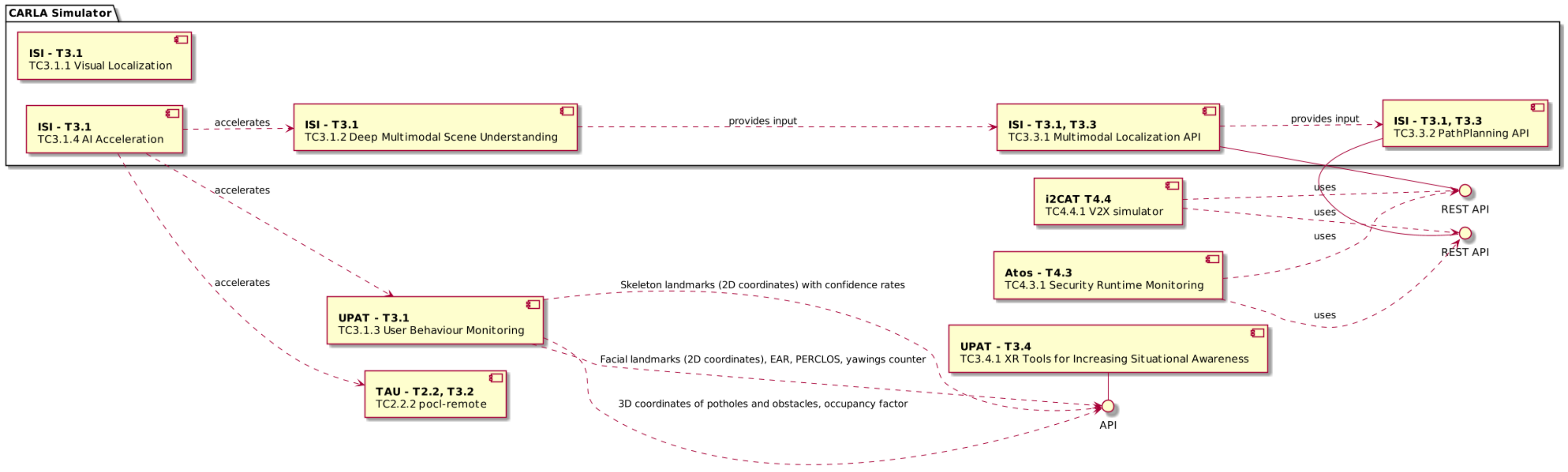
CARLA Simulator
ISI - T3.1
TC3.1.1 Visual Localization
ISI - T3.1
TC3.1.4 AI Acceleration
accelerates
ISI - T3.1
TC3.1.2 Deep Multimodal Scene Understanding
provides input
ISI - T3.1, T3.3
TC3.3.1 Multimodal Localization API
provides input
ISI - T3.1, T3.3
TC3.3.2 PathPlanning API
i2CAT T4.4
TC4.4.1 V2X simulator
uses
uses
uses
REST API
REST API
accelerates
accelerates
Atos - T4.3
TC4.3.1 Security Runtime Monitoring
uses
Skeleton landmarks (2D coordinates) with confidence rates
UPAT - T3.1
TC3.1.3 User Behaviour Monitoring
UPAT - T3.4
TC3.4.1 XR Tools for Increasing Situational Awareness
Facial landmarks (2D coordinates), EAR, PERCLOS, yawings counter
TAU - T2.2, T3.2
TC2.2.2 pocl-remote
3D coordinates of potholes and obstacles, occupancy factor
API

Figure 3 - Interfaces of CARLA-integrated components

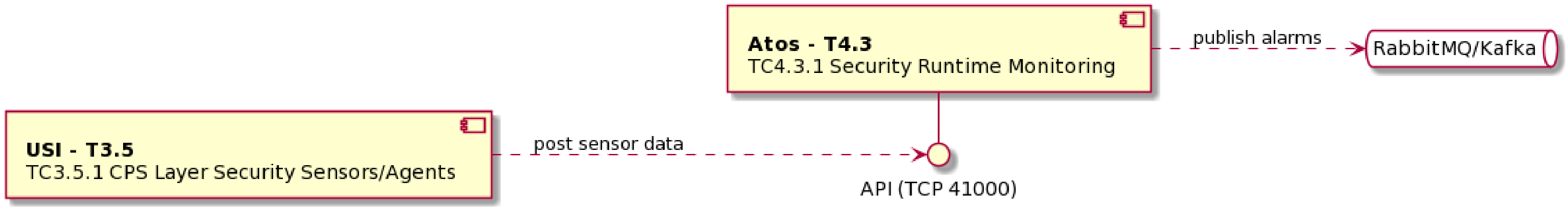

Figure 4 - Interfaces of Security Runtime Monitoring

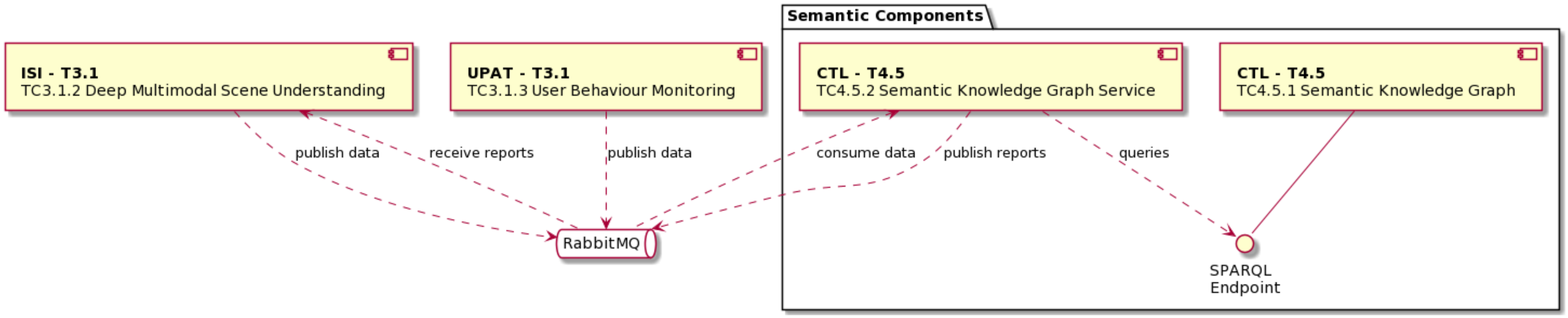

Figure 5 - Interfaces of Semantic components

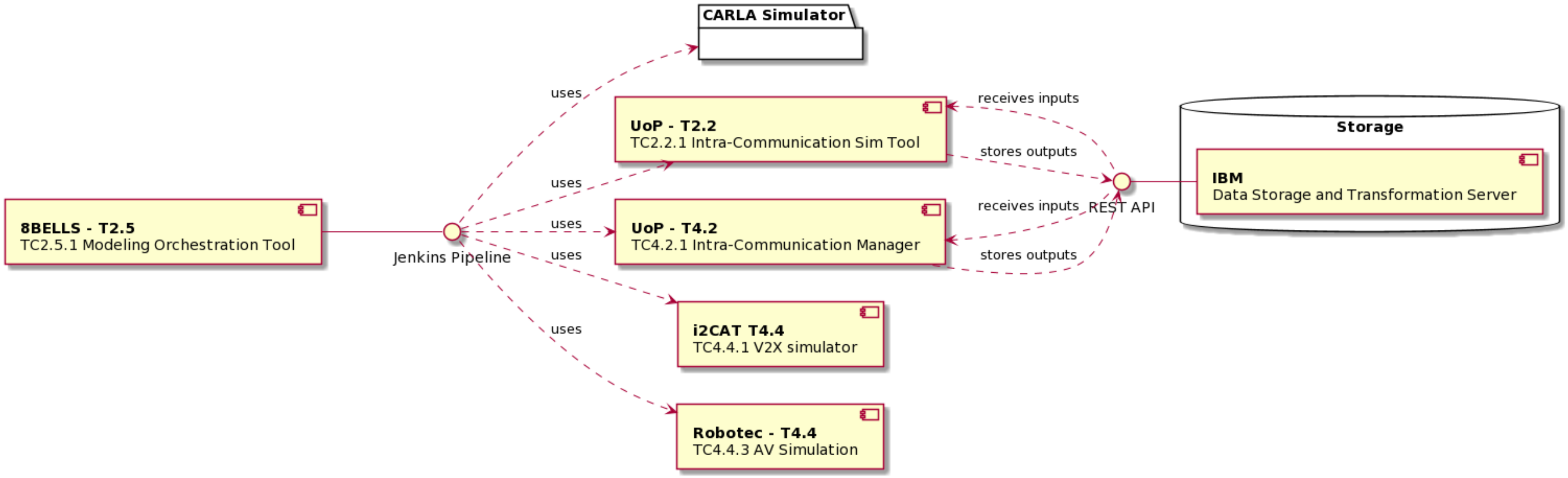

Figure 6 - Interfaces of simulators

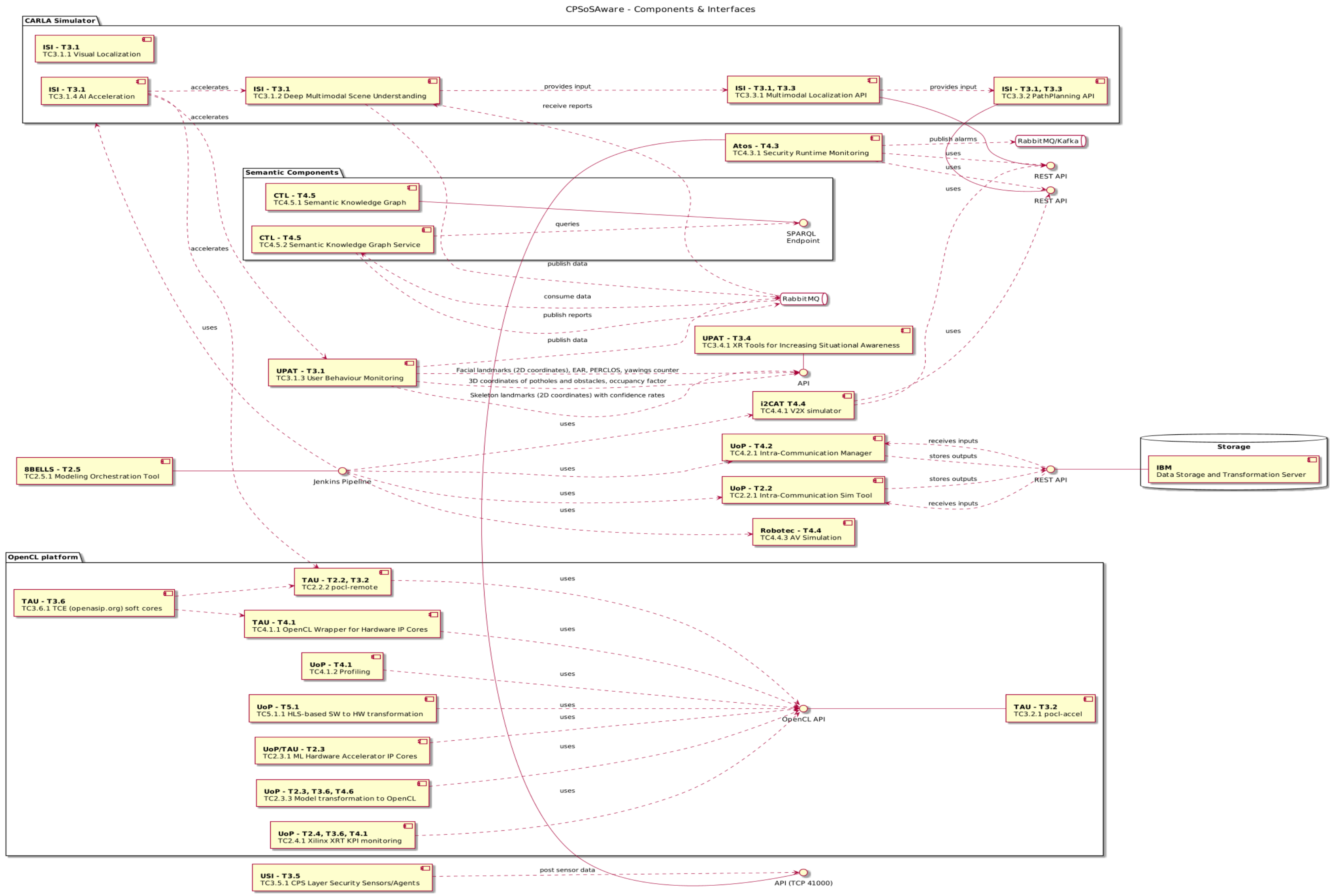
CPSoSAware - Components & Interfaces
CARLA Simulator
ISI - T3.1 TC3.1.1 Visual Localization
ISI - T3.1 TC3.1.4 AI Acceleration
accelerates
ISI - T3.1 TC3.1.2 Deep Multimodal Scene Understanding
provides input
receive reports
ISI - T3.1, T3.3 TC3.3.1 Multimodal Localization API
provides input
ISI - T3.1, T3.3 TC3.3.2 PathPlanning API
accelerates
Atos - T4.3 TC4.3.1 Security Runtime Monitoring
publish alarms
RabbitMQ/Kafka
uses
uses
uses
REST API
REST API
Semantic Components
CTL - T4.5 TC4.5.1 Semantic Knowledge Graph
CTL - T4.5 TC4.5.2 Semantic Knowledge Graph Service
queries
SPARQL Endpoint
accelerates
publish data
consume data
publish reports
RabbitMQ
uses
publish data
uses
UPAT - T3.4 TC3.4.1 XR Tools for Increasing Situational Awareness
UPAT - T3.1 TC3.1.3 User Behaviour Monitoring
Facial landmarks (2D coordinates), EAR, PERCLOS, yawings counter
3D coordinates of potholes and obstacles, occupancy factor
Skeleton landmarks (2D coordinates) with confidence rates
API
i2CAT T4.4 TC4.4.1 V2X simulator
uses
8BELLS - T2.5 TC2.5.1 Modeling Orchestration Tool
Jenkins Pipeline
uses
uses
uses
UoP - T4.2 TC4.2.1 Intra-Communication Manager
receives inputs
stores outputs
UoP - T2.2 TC2.2.1 Intra-Communication Sim Tool
stores outputs
receives inputs
REST API
Storage
IBM Data Storage and Transformation Server
Robotec - T4.4 TC4.4.3 AV Simulation
OpenCL platform
TAU - T3.6 TC3.6.1 TCE (openasip.org) soft cores
TAU - T2.2, T3.2 TC2.2.2 pocl-remote
TAU - T4.1 TC4.1.1 OpenCL Wrapper for Hardware IP Cores
UoP - T4.1 TC4.1.2 Profiling
UoP - T5.1 TC5.1.1 HLS-based SW to HW transformation
UoP/TAU - T2.3 TC2.3.1 ML Hardware Accelerator IP Cores
UoP - T2.3, T3.6, T4.6 TC2.3.3 Model transformation to OpenCL
UoP - T2.4, T3.6, T4.1 TC2.4.1 Xilinx XRT KPI monitoring
uses
uses
uses
uses
uses
uses
uses
OpenCL API
TAU - T3.2 TC3.2.1 pocl-accel
USI - T3.5 TC3.5.1 CPS Layer Security Sensors/Agents
post sensor data
API (TCP 41000)

Figure 7 - Overview of system interfaces (link for larger image)

4 Requirement View

4.1 System Requirement Monitoring Methodologies

Building upon the efforts presented in D1.3, the system requirement reference document has been updated and enriched with new fields to allow the efficient documentation and monitoring of the requirements' status. To this end, a new requirement specification template has been circulated to technical partners, for them to populate with information regarding the fulfilment status, the target fulfilment phase and relevant deliverable, and the requirement priority.

For the latter, we have adopted the MoSCoW prioritization methodology [1] that classifies requirements as *must-have* (critical requirements), *should-have* (important but not absolutely necessary), *could-have* (desirable but not necessary) and *won't-have* (least-critical or not appropriate).

Table 3 presents the fields of the new specification template, the format of expected responses and the lists of predefined values (where applicable).

Table 3 - Explanation of fields in the requirement specification template

Field	Explanation	Format	Values
Req. ID	The ID of the requirement	A unique ID to be typed manually. IDs should encode the TC they refer to. Suffix *R* indicates a functional requirement, while suffix *NFR* indicates a non-functional requirement.	
Description	An unambiguous requirement description	Free text	
Type	Identifies the type of the requirement, i.e., whether it is a functional requirement, security requirement or non-functional requirement (usability, reliability, efficiency, maintainability), integration requirement, etc.	Selection list	• Functionality • Usability • Reliability • Efficiency • Maintainability • Integration • Security • Functionality & Security • Functionality & Integration
Source	Defines how the requirement was identified End user: the system requirement has been elicited to satisfy a user requirement. DoA: The system requirement has been extracted from the Description of Action.	Selection list	• End user • DoA • End User & DoA
WP	The WP responsible for the effort to satisfy the requirement.	Selection list	WP1-WP8

Target Component	Indicates the Technical Component that is related to this system requirement.	Selection list	• TC2.1.1 • TC2.2.1 • Etc.
Target Phase	Indicates the expected system version where this requirement will have been addressed. System phases are associated to the CPSoSaware milestones, as described in the DoA, and the corresponding project months.	Selection list	• MS1 - Preliminary use cases and evaluation metrics - M12 • MS2 - CPSoSaware specifications and architecture - M24 • Etc.
Priority	Indicates how important the requirement is for the CPSoSaware system and its objectives. The priority level is defined using the MoSCoW technique.	Selection list	• M(ust) • S(hould) • C(ould) • W(on't)
Author	The partner suggesting the requirement.	Selection list	• ISI • I2CAT • IBM • Etc.
How addressed	Short description of how the requirement has been addressed. This is used for recording and monitoring progress of each requirement.	Free text	
Reported in	The deliverable(s) where the effort to address this requirement has been reported.	Free text	
Status	The fulfilment status of the requirement.	Selection list	• Not achieved yet • Partially achieved • Achieved • Rejected • Obsolete • Redundant

4.2 Current Status of System Requirements

Based on the information collected with the use of the latest requirement specification templates (see subsection 4.1), the following table lists the system requirements and their current status (as of M24 of the project).

Table 4 - Status of system requirements

Req. ID	Description	Type	Source	WP	Target comp.	Target phase	Priority	Author	How addressed	Reported in	Status
TC2.1.1.R1	Use an events list to register events	Functionality	End User & DoA	WP2	TC2.1.1 Data Collection Module	MS4 - First version of User Environment and communication Models - M12	S(hould)	8BELLS	The data collection was performed using google forms	D2.1	Redundant
TC2.1.1.R2	Be able to demonstrate in graphical way the HF models (that can be initially designed in UML format)	Functionality	End User & DoA	WP2	TC2.1.1 Data Collection Module	MS4 - First version of User Environment and communication Models - M12	S(hould)	8BELLS	The data collection was performed using google forms	D2.1	Redundant
TC2.1.1.R3	Use statistics to track important changes in variables	Functionality	End User & DoA	WP2	TC2.1.1 Data Collection Module	MS4 - First version of User Environment and communication Models - M12	S(hould)	8BELLS	The data collection was performed using google forms	D2.1	Redundant
TC2.1.1.R4	Should provide library of human metrics	Functionality	End User & DoA	WP2	TC2.1.1 Data Collection Module	MS4 - First version of User Environment and communication Models - M12	S(hould)	8BELLS	The data collection was performed using google forms	D2.1	Redundant
TC2.1.1.R5	Should be able to display in a graphical way these metrics	Functionality	End User & DoA	WP2	TC2.1.1 Data Collection Module	MS4 - First version of User Environment and communication Models - M12	S(hould)	8BELLS	The data collection was performed using google forms	D2.1	Redundant
TC2.1.1.R6	The operator should be able to query the Data Collection data metrics	Functionality	End User & DoA	WP2	TC2.1.1 Data Collection Module	MS4 - First version of User Environment and communication Models - M12	S(hould)	8BELLS	The data collection was performed using google forms	D2.1	Redundant

TC2.1.1.R7	The operator will be able to input data from a CSV/Spreadsheet into the DCM	Functionality	End User & DoA	WP2	TC2.1.1 Data Collection Module	MS4 - First version of User Environment and communication Models - M12	C(ould)	8BELLS	The data collection was performed using google forms	D2.1	Redundant
TC2.1.1.R8	The system shall ensure the confidentiality and integrity of the data being transmitted in the system	Security	End User & DoA	WP2	TC2.1.1 Data Collection Module	MS4 - First version of User Environment and communication Models - M12	M(ust)	8BELLS	The data collection was performed using google forms	D2.1	Redundant
TC2.1.1.R9	The system shall ensure the availability of its services to the relevant stakeholders	Reliability	End User & DoA	WP2	TC2.1.1 Data Collection Module	MS4 - First version of User Environment and communication Models - M12	S(hould)	8BELLS	The data collection was performed using google forms	D2.1	Redundant
TC2.1.1.NFR1	The DCM should scale automatically to meet the demand of new DCM metric data	Efficiency	End User & DoA	WP2	TC2.1.1 Data Collection Module	MS4 - First version of User Environment and communication Models - M12	S(hould)	8BELLS	The data collection was performed using google forms	D2.1	Redundant
TC2.1.1.NFR2	The DCM should provide a secure housing of The metrics data	Security	End User & DoA	WP2	TC2.1.1 Data Collection Module	MS4 - First version of User Environment and communication Models - M12	M(ust)	8BELLS	The data collection was performed using google forms	D2.1	Redundant
TC2.2.1.R1	The user should be able to feed the simulator with specific network scenario configuration	Functionality	End User & DoA	WP2	TC2.2.1 Intra-Communication Sim Tool	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Scenarios and network configuration are fed to simulator using message-driven framework.	D4.2	Achieved
TC2.2.1.R2	The tool will be able to record the simulation results in log files	Functionality	End User & DoA	WP2	TC2.2.1 Intra-Communication Sim Tool	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Simulation results are extracted in trace files	D4.2	Achieved

TC2.2.1.R3	The tool will be able to process the log files and extract the evaluation results of the simulation	Functionality	End User & DoA	WP2	TC2.2.1 Intra-Communication Sim Tool	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Simulation trace files are processed and presented in web-based dashboard (based on Grafana)	D4.2	Achieved
TC2.2.1.R4	The simulation outcomes will be able to be indexed in database and visualized (e.g., Prometheus/Grafana, ElasticSearch/Kibana)	Functionality	End User & DoA	WP2	TC2.2.1 Intra-Communication Sim Tool	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Simulation trace files are processed and presented in web-based dashboard (based on Grafana)	D4.2	Achieved
TC2.2.1.R5	The evaluation results will be formulated and fed back to the input of the tool to perform optimizations through iterations	Functionality	End User & DoA	WP2	TC2.2.1 Intra-Communication Sim Tool	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Software running on raw simulation data are aggregating results and feedback the simulator.	D4.2	Partially achieved
TC2.2.1.NFR1	The tool should be able to scale according to the load	Functionality	End User & DoA	WP2	TC2.2.1 Intra-Communication Sim Tool	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	UOP	Simulation jobs are scheduled and served asynchronously based on the available resources	D4.2	Achieved
TC2.2.1.NFR2	Adoption of microservices paradigm (e.g. containerization)	Functionality	End User & DoA	WP2	TC2.2.1 Intra-Communication Sim Tool	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	UOP	The NS3 Simulation as a Service is fully containerized deployed in a Kubernetes cluster	D4.2	Achieved
TC2.2.1.NFR3	Authentication/authorization schemes will be supported	Functionality & Integration	End User & DoA	WP2	TC2.2.1 Intra-Communication Sim Tool	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	UOP	OAUTH2 schemes will be employed	D4.2	Achieved

TC2.2.1.NFR4	The tool will expose well defined APIs to allow 3rd party services to integrate	Usability	End User & DoA	WP2	TC2.2.1 Intra-Communication Sim Tool	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	RESTFul APIs will be designed and exposed for integration with 3rd party components	D4.2	Partially achieved
TC2.2.1.NFR5	The tool will be available online	Usability	End User & DoA	WP2	TC2.2.1 Intra-Communication Sim Tool	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	UOP	The tool has beed designed to be completely web based and accessible following the microservices architectural paradigm	D4.2	Achieved
TC2.2.2.R1	Provide access to all OpenCL-supported devices in a network distributed platform from a single host application.	Functionality	End User & DoA	WP2	TC2.2.2 pocl-remote	MS8 - Final CPHs Architecture Design and Implementation - M36	M(ust)	TAU	Work mostly imported from another project's results, adapted to the CPSoSAware needs.	D3.2 (due M28)	Partially achieved
TC2.2.2.R2	Support peer-to-per synchronization of devices without host-application round trips.	Efficiency	End User & DoA	WP2	TC2.2.2 pocl-remote	MS8 - Final CPHs Architecture Design and Implementation - M36	S(hould)	TAU	In progress.	D3.2 (due M28)	Partially achieved
TC2.2.2.NFR1	At most 15% overhead in latency on top of the unavoidable network latencies.	Efficiency	End User & DoA	WP2	TC2.2.2 pocl-remote	MS8 - Final CPHs Architecture Design and Implementation - M36	S(hould)	TAU	In progress.	D3.2 (due M28)	Partially achieved
TC2.2.2.NFR2	Can utilize 80% of the theoretical bandwidth for buffer transfers.	Efficiency	End User & DoA	WP2	TC2.2.2 pocl-remote	MS8 - Final CPHs Architecture Design and Implementation - M36	S(hould)	TAU	In progress.	D3.2 (due M28)	Partially achieved

TC2.2.3.R1	Secure identification of CPSs			WP2	TC2.2.3 Slice Manager			I2CAT			Obsolete
TC2.2.3.R2	Monitors and assures the behavior and performance of the various slices through collecting network function and infrastructure data			WP2	TC2.2.3 Slice Manager			I2CAT			Obsolete
TC2.2.3.R3	Slice automation and orchestration			WP2	TC2.2.3 Slice Manager			I2CAT			Obsolete
TC2.2.3.R4	Need to support slice modelling by changing various network functions, connection and links to create specific network services			WP2	TC2.2.3 Slice Manager			I2CAT			Obsolete
TC2.2.3.NFR1	Delay of service instantiate			WP2	TC2.2.3 Slice Manager			I2CAT			Obsolete
TC2.2.3.NFR2	Service recovery failure and service continuity			WP2	TC2.2.3 Slice Manager			I2CAT			Obsolete
TC2.2.4.R1	Potentially able to levarage the features and functionality of OSM MANO			WP2	TC2.2.4 LightEdge			I2CAT			Obsolete
TC2.2.4.R2	Compling with the Cloud native solutions and allowing the contenarized edge application			WP2	TC2.2.4 LightEdge			I2CAT			Obsolete

TC2.2.4.R3	Capable of supporting the Local breakout for enterprise application			WP2	TC2.2.4 LightEdge			I2CAT			Obsolete
TC2.2.4.NFR1	Handling the user mobility			WP2	TC2.2.4 LightEdge			I2CAT			Obsolete
TC2.2.4.NFR2	Scalability between the LightEdge enabled MEC servers			WP2	TC2.2.4 LightEdge			I2CAT			Obsolete
TC2.2.4.NFR3	Service failure recovery			WP2	TC2.2.4 LightEdge			I2CAT			Obsolete
TC2.3.1.R1	Accelerate DNN inference in comparison to software running in ARM.	Efficiency	End User & DoA	WP2	TC2.3.1 ML Hardware Accelerator IP Cores	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Implementation on FPGA equipped with ARM processing cores	D4.1	Partially achieved
TC2.3.1.NFR1	At least 20% faster 8b convolutions achieved.	Efficiency	End User & DoA	WP2	TC2.3.1 ML Hardware Accelerator IP Cores	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Implemented on FPGA	D4.3	Partially achieved
TC2.3.2.R1	Confidentiality: The components should provide cryptography services for popular Public and Private encryption algorithms). Support for Public Key Infrastructure should be possible)	Security	End User & DoA	WP2	TC2.3.2 Security Accelerators for CPS security agents/sensors	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	USI	Components have been created to achieve this goal in hardware and/or software	D2.3 and D3.5	Partially achieved
TC2.3.2.R2	Data Integrity: The components should provide cryptography	Security	End User & DoA	WP2	TC2.3.2 Security Accelerators for CPS security agents/sensors	MS7 - Intermediate CPHs Architecture Design and	M(ust)	USI	Components have been created to achieve this goal in hardware and/or software	D2.3 and D3.5	Partially achieved

	services for popular message Integrity Mechanisms include MAC functions, Digital Signature and Authenticated Encryption					Implementation - M24					
TC2.3.2.R3	Authentication: The components should be able to provide authentication services including message authentication, machine to machine (M2M) authentication.	Security	End User & DoA	WP2	TC2.3.2 Security Accelerators for CPS security agents/sensors	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	USI	Components have been created to achieve this goal in hardware and/or software	D2.3 and D3.5	Partially achieved
TC2.3.2.NFR1	Resistance against security attacks (side channel attacks)	Security	DoA	WP2	TC2.3.2 Security Accelerators for CPS security agents/sensors	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	USI	The componenets have been strengthen tp provide resistnace against Side channel attacks	D2.3 and D3.5	Partially achieved
TC2.3.2.NFR2	Strong Cryptographic Strength	Security	End User & DoA	WP2	TC2.3.2 Security Accelerators for CPS security agents/sensors	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	USI	The components offer quantum safe level of security which is the considerably strong security	D2.3 and D3.5	Partially achieved
TC2.3.2.NFR3	Reliability-Fault tolerence	Reliability	DoA	WP2	TC2.3.2 Security Accelerators for CPS security agents/sensors	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	USI	Resistance against fault injection attacks have been introduced in some of the developed components	D2.3 and D3.5	Partially achieved
TC2.3.2.NFR4	Efficiency (response time)	Efficiency	End User & DoA	WP2	TC2.3.2 Security Accelerators for	MS7 - Intermediate CPHs Architecture Design and	S(hould)	USI	The component architecture using HSL tools or directly	D2.3 and D3.5	Partially achieved

					CPS security agents/sensors	Implementation - M24			hardware description language optimization has been made considerably efficient in terms of speed		
TC2.3.2.NFR5	Efficiency (constrained memory and chip covered area resources)	Efficiency	End User & DoA	WP2	TC2.3.2 Security Accelerators for CPS security agents/sensors	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	USI	Some of the developed components are designed as lightweight architectures (e.g. lightweight cryptography schemes) that consume minimal number of utilized resources	D2.3 and D3.5	Partially achieved
TC2.3.2.NFR6	Flexibility	Usability	DoA	WP2	TC2.3.2 Security Accelerators for CPS security agents/sensors	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	USI	The developed components have been parameterized so as to be adjustable to various security levels	D2.3 and D3.5	Partially achieved
TC2.3.2.NFR7	Interoperability	Usability	DoA	WP2	TC2.3.2 Security Accelerators for CPS security agents/sensors	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	USI	The developed components are adaptable to any CPSoSaware cybersecurity scenario.	D2.3 and D3.5	Partially achieved
TC2.3.3.R1	Profiling	Efficiency	End User & DoA	WP2	TC2.3.3 Model transformation to openCL	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Driver State Monitoring (DSM) application based on DEST library. Profiled to recognize the candidate functions for HW acceleration	D2.2	Partially achieved
TC2.3.3.R2	HLS based SW to HW Transformation	Efficiency	End User & DoA	WP2	TC2.3.3 Model transformation to openCL	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	HW synthesis performed in Xilinx Vitis	D2.2	Partially achieved
TC2.3.3.NFR1	Development of HW-SW Library	Efficiency	End User & DoA	WP2	TC2.3.3 Model transformation to openCL	MS7 - Intermediate CPHs Architecture Design and	M(ust)	UOP	FPGA implementations of DSM components as well as other HSL	D3.6	Partially achieved

	with reliable Components.					Implementation - M24			components used to populate this library		
TC2.4.1.R1	Accelerate DNN inference in comparison to software running in ARM.	Efficiency	End User & DoA	WP2	TC2.4.1 Xilinx XRT KPI monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	A CNN for handwritten character recognition employed as a use case	D3.2	Partially achieved
TC2.4.1.R2	HLS based SW to HW Transformation	Functionality	End User & DoA	WP2	TC2.4.1 Xilinx XRT KPI monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	HLS implementation of CNN for handwritten character recognition	D3.2	Partially achieved
TC2.4.1.R3	Commissioning: The component should be able to collect hardware bitstreams IP Cores and download them on the FPGA fabric of a Multiprocessor System on Chip FPGA board.	Functionality	End User & DoA	WP2	TC2.4.1 Xilinx XRT KPI monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Vitis XRT is used to support HW configuration bitstream download to FPGA	D2.3	Partially achieved
TC2.4.1.R4	Reconfigurability: The components should be able to reconfigure the commissioned hardware IP Cores on the FPGA fabric of a TC4.6.1.R3 Multiprocessor System on Chip FPGA board and replace existing hardware IP Cores.	Functionality	End User & DoA	WP2	TC2.4.1 Xilinx XRT KPI monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Multiple models for face alignment in the DSM module have been trained and can be employed in dynamic reconfiguration of HW kernels according to environmental conditions or other inputs	D2.3	Partially achieved

TC2.4.1.R5	Removal: The component should be able to remove existing hardware IP Cores in the FPGA fabric of a Multiprocessor System on Chip (MPSoS) FPGA board.	Functionality	End User & DoA	WP2	TC2.4.1 Xilinx XRT KPI monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Dynamic reconfiguration of HW kernels implies HW kernel removal	D2.3	Partially achieved
TC2.4.1.R6	Accessibility: The component should be able to communicate with the model-based design mechanism of the CPSoSaware layer in order to deploy hardware IP Cores in the MPSoC board. (TC4.6.1.R5)	Functionality	End User & DoA	WP2	TC2.4.1 Xilinx XRT KPI monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Implemented as part of the dynamic reconfiguration of HW kernels	D2.3	Partially achieved
TC2.4.1.NFR1	Development of HW-SW Library with reliable Components	Efficiency	End User & DoA	WP2	TC2.4.1 Xilinx XRT KPI monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	FPGA implementations of DSM components as well as other HSL components used to populate this library	D3.6	Partially achieved
TC2.4.1.NFR2	The component should be able to handle efficiently the configuration updates and resolve any possible dependencies	Efficiency	End User & DoA	WP2	TC2.4.1 Xilinx XRT KPI monitoring	MS6 - Preliminary CPHs Architecture Design and Implementation - M12	C(ould)	UOP	Some indications about dependencies have been highlighted during profiling	D3.6	Partially achieved
TC2.4.1.NFR3	The component should be able to provide integrity validation method in both ends (e.g.	Reliability	End User & DoA	WP2	TC2.4.1 Xilinx XRT KPI monitoring	MS6 - Preliminary CPHs Architecture Design and Implementation - M12	C(ould)	UOP	In the DSM module, validation of the face boundaries has been applied	D3.6	Partially achieved

	hashes of the transferred payloads)										
TC2.4.1.NFR4	The component should be aware of the commissioning process' status and handle failures (e.g. rollback to previous versions)	Reliability	End User & DoA	WP2	TC2.4.1 Xilinx XRT KPI monitoring	MS6 - Preliminary CPHs Architecture Design and Implementation - M12	C(ould)	UOP	Methods to roll back to a valid state will be investigated where applicable	D3.6	Not achieved yet
TC2.5.1.R1	User-driven orchestration control events to initiate orchestration.	Functionality & integration	End User & DoA	WP2	TC2.5.1 Modelling Orchestration Tool	MS5 - Final version of User Environment communication Models - M24	M(ust)	8BELLS	Developed an orchestration environment with an embedded UI	D2.2	Achieved
TC2.5.1.R2	Autonomic model-driven orchestration control events by models	Functionality & integration	DoA	WP2	TC2.5.1 Modelling Orchestration Tool	MS5 - Final version of User Environment communication Models - M24	C(ould)	8BELLS	Simulation model development is not related to the orchestration process	D2.2	Not achieved yet
TC2.5.1.R3	Integrate existing CPS modelling-simulation tools	Integration	DoA	WP2	TC2.5.1 Modelling Orchestration Tool	MS5 - Final version of User Environment communication Models - M24	S(hould)	8BELLS	Carla, NS3, SUMO, RoSi, Robotec V2X, and Manufacturing Simulators are integrated to the orchestrations tool	D2.2	Achieved
TC2.5.1.NFR1	Minimize centralized control of the orchestration	Functionality	End User & DoA	WP2	TC2.5.1 Modelling Orchestration Tool	MS5 - Final version of User Environment communication Models - M24	S(hould)	8BELLS	The orchestrator applications can be use by whoever has the credentials to control the simulations	D2.2	Achieved
TC2.5.1.NFR2	Reliable and secure autonomic operations	Functionality & security	DoA	WP2	TC2.5.1 Modelling Orchestration Tool	MS5 - Final version of User Environment communication Models - M24	S(hould)	8BELLS	Every tool is running on an individual safe and secure environment	D2.2	Achieved
TC3.1.1.R1	Trajectory of vehicle generated by CARLA.	Integration	End User & DoA	WP3	TC3.1.1 Visual Localization	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Low-cost Methods for Visual SLAM that can	D3.1	Partially achieved

									run efficiently on edge devices		
TC3.1.1.R2	Database of geo-tagged images available.	Functionality & integration	End User & DoA	WP3	TC3.1.1 Visual Localization	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Synthetic images combined with ground truth position extracted by the Carla simulation environment	D3.1	Partially achieved
TC3.1.1.NFR1	Minimize the computational time of visual search in the database	Efficiency	End User & DoA	WP3	TC3.1.1 Visual Localization	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	A bag-of-words approach for fast image search and retrieval	D3.1	Partially achieved
TC3.1.2.R1	Availability of RGBD and point cloud data	Functionality & integration	End User & DoA	WP3	TC3.1.2 Deep Multimodal Scene Understanding	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Synthetic RGBD and lidar data extracted by the Carla simulation environment	D3.1	Achieved
TC3.1.2.R2	Camera mapping strategy and LIDAR processing approach for effective data fusion	Functionality & integration	End User & DoA	WP3	TC3.1.2 Deep Multimodal Scene Understanding	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Low cost multi modal fusion approach for integrating visual and lidar sensor data	D3.1	Not achieved yet
TC3.1.2.R3	Post-processing semantic analysis functionality	Functionality	End User & DoA	WP3	TC3.1.2 Deep Multimodal Scene Understanding	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Effective image and point cloud processing for semantic segmentation	D3.1	Not achieved yet
TC3.1.2.NFR1	Real-time execution	Efficiency	End User & DoA	WP3	TC3.1.2 Deep Multimodal Scene Understanding	MS2 - CPSoSaware specifications and architecture - M24	M(ust)	ISI	Algorithms with bounded execution time deployed on accelerated hardware	D3.1	Partially achieved
TC3.1.2.NFR2	Efficient semantic representation to reduce required training data	Efficiency	End User & DoA	WP3	TC3.1.2 Deep Multimodal Scene Understanding	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Effective image and point cloud processing for semantic segmentation	D3.1	Partially achieved
TC3.1.3.R1	Pre-trained model of faces for the real-time face	Maintainability	DoA	WP3	TC3.1.3 User Behaviour Monitoring	MS5 - Final version of User Environment	S(hould)	UPAT	CNN algorithm that extracts facial landmarks	D3.1	Achieved

	recognition and face tracking via markers					communication Models - M24					
TC3.1.3.R2	Continuously recording of the driver's face	Functionality	End User & DoA	WP3	TC3.1.3 User Behaviour Monitoring	MS5 - Final version of User Environment communication Models - M24	M(ust)	UPAT	Real time implementation using camera	D3.1	Partially achieved
TC3.1.3.R3	Optimization of algorithms for running in real-time	Efficiency	DoA	WP3	TC3.1.3 User Behaviour Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	UPAT	Parameterization of the algorithm to run real time	D3.1	Not achieved yet
TC3.1.3.R4	Decision making based on the drowsiness level, indicating the appropriate warning signs.	Usability	End User & DoA	WP3	TC3.1.3 User Behaviour Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UPAT	Metrics and SoA rules that indicate the drowsiness	D3.1	Achieved
TC3.1.3.R5	Continuous monitoring and recording of driver's pulse rate from a wearable device.	Functionality	End User & DoA	WP3	TC3.1.3 User Behaviour Monitoring	MS5 - Final version of User Environment communication Models - M24	C(ould)	UPAT	Use of wearable devices to monitor biometrics of the drivers	D3.1	Not achieved yet
TC3.1.3.NFR1	Computational efficiency	Efficiency	End User & DoA	WP3	TC3.1.3 User Behaviour Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	UPAT	Parameterization of the algorithm to be computationally efficient	D3.1	Not achieved yet
TC3.1.3.NFR2	Security of the driver or human operator	Security	End User & DoA	WP3	TC3.1.3 User Behaviour Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UPAT	Continuously monitoring the users; status and behavior	D3.1	Partially achieved
TC3.1.3.NFR3	Maximization of the situational awareness	Functionality & security	End User & DoA	WP3	TC3.1.3 User Behaviour Monitoring	MS7 - Intermediate CPHs Architecture Design and	M(ust)	UPAT	Visualization of potholes and obstacles before the drivers can see them	D3.1	Partially achieved

						Implementation - M24					
TC3.1.3.NFR4	Robustness under different light conditions	Reliability	End user	WP3	TC3.1.3 User Behaviour Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	UPAT	Use of point clouds that their geometry is not affected by light conditions	D3.1	Not achieved yet
TC3.1.4.R1	Pre-trained DCNN available: Original DCNN model, pre-trained for the target application, available in ONNX, or MATLAB, or TF format.	Usability	End User & DoA	WP3	TC3.1.4 AI Acceleration	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Employ software tools which enable the transformation of TF format to MATLAB and ONNX	D3.1	Partially achieved
TC3.1.4.R2	Data availability: Training/validation dataset, for the target application, available for retraining/finetuning purposes.	Functionality	End User & DoA	WP3	TC3.1.4 AI Acceleration	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Extract training dataset (images, point cloud, positions) from CARLA simulation environment	D3.1	Achieved
TC3.1.4.R3	Accelerated DCNN runtime functionality: Availability of parameter-sharing enabled convolutional layer implementation.	Functionality	DoA	WP3	TC3.1.4 AI Acceleration	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	ISI	Well known model compression and acceleration methods will be used	D3.1	Partially achieved
TC3.1.4.NFR1	Accelerated model accuracy within user specifications.	Functionality	DoA	WP3	TC3.1.4 AI Acceleration	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	ISI	Well known model compression and acceleration methods will be used	D3.1	Partially achieved
TC3.1.4.NFR2	Minimize accelerated DCNN	Functionality & integration	DoA	WP3	TC3.1.4 AI Acceleration	MS7 - Intermediate CPHs Architecture Design and	S(hould)	ISI	Well known model compression and	D3.1	Partially achieved

	model storage space.					Implementation - M24			acceleration methods will be used		
TC3.1.4.NFR3	Minimize accelerated DCNN model inference time execution.	Functionality & integration	DoA	WP3	TC3.1.4 AI Acceleration	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	ISI	Well known model compression and acceleration methods will be used	D3.1	Partially achieved
TC3.2.1.R1	Must be able to support at least OpenCL 1.2 based command queues on the AlmaIF	Integration	DoA	WP3	TC3.2.1 pocl-accel	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	TAU	Implemented and reported. Publication presented in NorCAS 2021.	D2.3	Achieved
TC3.2.1.NFR1	Driver overhead less than 1%	Efficiency	DoA	WP3	TC3.2.1 pocl-accel	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	TAU	Implemented and reported.	D2.3	Achieved
TC3.3.1.R1	Trajectories of vehicles: CARLA simulator will generate the trajectories of vehicles moving in a city.	Functionality	DoA	WP3	TC3.3.1 Multimodal Localization API	MS10 - Final version of the Simulation Software - M24	S(hould)	ISI	Vehicle trajectories generated by CARLA simulator are extracted to csv files	D3.3	Achieved
TC3.3.1.R2	Measurement availability: It is assumed that absolute position and range measurements from GPS and LIDAR sensor will always be available.	Functionality	DoA	WP3	TC3.3.1 Multimodal Localization API	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Measurements will be produced from the ground truth positions degraded by Gaussian noise	D3.3	Achieved
TC3.3.1.R3	Cooperation: Multi-modal fusion will be performed in a collaborating	Functionality	DoA	WP3	TC3.3.1 Multimodal Localization API	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Cooperation will be established within a fixed communication range	D3.3	Achieved

	manner, by representing the VANET as a graph.										
TC3.3.1.NFR1	Measurements degraded by Gaussian noise.	Functionality	DoA	WP3	TC3.3.1 Multimodal Localization API	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Measurements will be produced from the ground truth positions degraded by Gaussian noise	D3.3	Achieved
TC3.3.1.NFR2	Exchange of measurements and estimation of locations before the new GPS measurement.	Functionality	DoA	WP3	TC3.3.1 Multimodal Localization API	MS2 - CPSoSaware specifications and architecture - M24	M(ust)	ISI	GPS updating time should be between 0.2 and 0.4 sec	D3.3	Achieved
TC3.3.2.R1	Location Logging Mechanism: The component should be able to collect logs of the node's position.	Functionality	End user	WP3	TC3.3.2 PathPlanning API	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Position log's update time between 0.2 and 0.4 sec	D3.3	Not achieved yet
TC3.3.2.R2	Control Error Logging Mechanism: The component should be able to collect logs related to the path planning control error.	Functionality	End user	WP3	TC3.3.2 PathPlanning API	MS10 - Final version of the Simulation Software - M24	C(ould)	ISI	Control log's update time between 0.2 and 0.4 sec	D3.3	Not achieved yet
TC3.3.2.R3	Execution Time: The component should be able to collect the measured time between update of sensor inputs till response to updated inputs for each node.	Functionality	End user	WP3	TC3.3.2 PathPlanning API	MS10 - Final version of the Simulation Software - M24	S(hould)	ISI	Address within the context of simulating visual sensors inside CARLA	D3.3	Not achieved yet

TC3.3.2.R4	Connectivity graph: The component should be able to store the nodes that are actively collaborate to optimize the path planning control.	Functionality	End user	WP3	TC3.3.2 PathPlanning API	MS10 - Final version of the Simulation Software - M24	S(hould)	ISI	Cooperation will be established within a fixed communication range	D3.3	Not achieved yet
TC3.3.2.R5	Awareness level: The component should be able to store the awareness level (SAL) metric.	Functionality	End User & DoA	WP3	TC3.3.2 PathPlanning API	MS10 - Final version of the Simulation Software - M24	S(hould)	ISI	In progress	D3.3	Not achieved yet
TC3.3.2.NFR1	Minimize centralized control	Usability		WP3	TC3.3.2 PathPlanning API	MS2 - CPSoSaware specifications and architecture - M24	C(ould)	ISI	Deploy distributed Laplacian based path planning solutions	D3.3	Not achieved yet
TC3.3.2.NFR2	Minimize collision risk	Usability		WP3	TC3.3.2 PathPlanning API	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Optimize the attained accuracy of cooperative awareness solution	D3.3	Not achieved yet
TC3.3.2.NFR3	Maximize fault tolerance	Reliability		WP3	TC3.3.2 PathPlanning API	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Optimize the attained accuracy of cooperative awareness solution	D3.3	Not achieved yet
TC3.3.2.NFR4	Maximize situational awareness	Efficiency		WP3	TC3.3.2 PathPlanning API	MS2 - CPSoSaware specifications and architecture - M24	S(hould)	ISI	Optimize the attained accuracy of cooperative awareness solution	D3.3	Not achieved yet
TC3.4.1.R1	Information streams regarding the task underway improving focus	Reliability	DoA	WP3	TC3.4.1 XR tools for increasing situational awareness	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	UPAT	In progress	D3.4	Not achieved yet
TC3.4.1.R2	Personalized reminders regarding other parallel or scheduled tasks significantly		DoA	WP3	TC3.4.1 XR tools for increasing situational awareness		W(on't)	UPAT		D3.4	Rejected

	improving response time										
TC3.4.1.R3	Notifications and visual aids regarding imminent dangers or accident-related factors (e.g., pothole and obstacle detection, or operator entering unsafe (robot's) zone)	Functionality & security	End User & DoA	WP3	TC3.4.1 XR tools for increasing situational awareness	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UPAT	When a dangerous situation appears then a corresponding notification will inform the users	D3.4	Partially achieved
TC3.4.1.R4	KPIs visualizing the effectiveness of the CPSoS functionality	Reliability	DoA	WP3	TC3.4.1 XR tools for increasing situational awareness	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	UPAT	The estimated KPIs of a functionality will be presented to the users	D3.4	Not achieved yet
TC3.4.1.R5	Cooperative situational awareness.Visualization and use of coalition information provided by other vehicles or interactive robots (e.g., highlighting of occluded vehicles and pedestrians)	Functionality & security	DoA	WP3	TC3.4.1 XR tools for increasing situational awareness	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UPAT	Use of coalition information provided by other vehicles	D3.4	Achieved
TC3.4.1.NFR1	Computational efficiency (real-time)	Efficiency	End User & DoA	WP3	TC3.4.1 XR tools for increasing situational awareness	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	UPAT	Parameterization of the algorithm to be computationally efficient	D3.4	Partially achieved

TC3.4.1.NFR2	User-friendly interface	Usability	End User & DoA	WP3	TC3.4.1 XR tools for increasing situational awareness	MS8 - Final CPHs Architecture Design and Implementation - M36	C(ould)	UPAT	The provided information will be easy to understand by the users		Not achieved yet
TC3.4.1.NFR3	Reliability and robustness of the provided awareness sign	Reliability	DoA	WP3	TC3.4.1 XR tools for increasing situational awareness	MS8 - Final CPHs Architecture Design and Implementation - M36	S(hould)	UPAT	We need to remove the false alarm situations	D3.4	Not achieved yet
TC3.4.1.NFR4	Improve situational awareness without disturbing the user's attention	Functionality	End User & DoA	WP3	TC3.4.1 XR tools for increasing situational awareness	MS8 - Final CPHs Architecture Design and Implementation - M36	M(ust)	UPAT	Provide intuitive and non-distractive information	D3.4	Not achieved yet
TC3.4.1.NFR5	Provide only useful information based on personalized user's preferences	Usability	End User & DoA	WP3	TC3.4.1 XR tools for increasing situational awareness		S(hould)	UPAT	Take into account the personal preferences of the users	D3.4	Not achieved yet
TC3.5.1.R1	Logging Mechanism: The component should be able to collect logs of event that take place in a CPS platform	Functionality & security	End User & DoA	WP3	TC3.5.1 CPS layer Security sensors/agents	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	USI	A security event logging mechanism has been implemented as a C code library and has been integrated in the CPSoSaware end node security device	D3.5	Partially achieved
TC3.5.1.R2	Data Integrity: The component should be able to ensure integrity of collected data that are forwarded to the CPSoSaware Runtime Monitoring System	Security	End User & DoA	WP3	TC3.5.1 CPS layer Security sensors/agents	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	USI	A data integrity mechanism based on symmetric and public key cryptography has been implemented in the CPSoSaware end node using a hardware security token	D3.5	Partially achieved
TC3.5.1.R3	Data Authenticity: The component should be able to ensure authenticity of collected data that are forwarded	Security	End User & DoA	WP3	TC3.5.1 CPS layer Security sensors/agents	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	USI	A data integrity mechanism based on symmetric and public key cryptography has been implemented in the CPSoSaware end	D3.5	Partially achieved

	to the CPSoSaware Runtime Monitoring System								node using a hardware security token		
TC3.5.1.R4	Detectability: The component should be able to detect simple anomalous events in the CPS system (e.g. related to false data injection, security attacks on the device and CPS network issues)	Security	End User & DoA	WP3	TC3.5.1 CPS layer Security sensors/agents	MS8 - Final CPHs Architecture Design and Implementation - M36	M(ust)	USI	In progress	D3.5	Partially achieved
TC3.5.1.R5	Secure channel communication: The component should be able to transmit in a secure and trusted way the collected logs to the CPSoSawre Runtime monitoring system. This can be manages through end to end secure communication	Functionality & security	End User & DoA	WP3	TC3.5.1 CPS layer Security sensors/agents	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	USI	A secure communication framework based on symmetric and public key cryptography (TLS1.3 based) has been implemented in the CPSoSaware end node using a hardware security token	D3.5	Partially achieved
TC3.5.1.NFR1	Efficiency (response time)	Efficiency	DoA	WP3	TC3.5.1 CPS layer Security sensors/agents	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	USI	The computationally demanding and slow security components that has been deployed in the CPS layer are accelerated through hardware means to match the efficiency goals	D3.5	Partially achieved

TC3.5.1.NFR2	Efficiency (constrained memory and chip covered area resources)	Efficiency	DoA	WP3	TC3.5.1 CPS layer Security sensors/agents	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	USI	The computationally demanding and slow security components that has been deployed in the CPS layer are accelerated through hardware means in an optimal way that minimizes the memory and chip covered areas usage	D3.5	Partially achieved
TC3.5.1.NFR3	Flexibility so that sensor components can be updated dynamically	Usability	DoA	WP3	TC3.5.1 CPS layer Security sensors/agents	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	USI	Security components are becoming flexible by using reconfigurable logic (FPGA) at the hardware level and algorithmic modifications at the software level (when needed)	D3.5	Partially achieved
TC3.5.1.NFR4	Interoperability so that sensors can be used in various different CPSs and both pilots	Usability	DoA	WP3	TC3.5.1 CPS layer Security sensors/agents	MS8 - Final CPHs Architecture Design and Implementation - M36	C(ould)	USI	The security sensors are generic and are not applicable to any single scenario. The Hardware Security Token deployed at the CPS level has a generic CLI environment (similar to the openSSL CLI) that is pilot and device agnostic	D3.5	Not achieved yet
TC3.5.1.NFR5	Trusted computation following security by design approach and use of trusted execution environments	Security	DoA	WP3	TC3.5.1 CPS layer Security sensors/agents	MS8 - Final CPHs Architecture Design and Implementation - M36	C(ould)	USI	The security components are resistant against various implementation (side channel attacks) thus thy can be considered trusted. Also, when applicable	D3.5	Not achieved yet

									(supported by the platform's hardware) trusted execution environments are used for the execution of the security functionality at the CPS level		
TC3.6.1.R1	Programmable co-processor for cases where hardware customization is useful, but runtime programmability is needed.	Functionality	DoA	WP3	TC3.6.1 TCE (openasip.org) soft cores	MS12 - Intermediate CPSoSAware End to End Platform and Application Design - M24	M(ust)	TAU	The base functionality works.	D2.3	Achieved
TC3.6.1.R2	Ability to execute at least two different tasks defined by switching the software binary only.	Functionality	DoA	WP3	TC3.6.1 TCE (openasip.org) soft cores	MS12 - Intermediate CPSoSAware End to End Platform and Application Design - M24	M(ust)	TAU	The base functionality works.	D2.3	Achieved
TC3.6.1.NFR1	Performance requirements are task/application specific. Overall, acceleration or improved energy-efficiency over similar software on a general purpose processor is required to justify an ASIP.	Efficiency	DoA	WP3	TC3.6.1 TCE (openasip.org) soft cores	MS8 - Final CPHs Architecture Design and Implementation - M36	S(hould)	TAU	Efficiency will be improved in T3.6.	D3.6	Partially achieved
TC4.1.1.R1	Ability to easily add IPs and co-processors to OpenCL platforms that are orchestrated from	Functionality & integration	DoA	WP4	TC4.1.1 OpenCL Wrapper for Hardware IP Cores	MS2 - CPSoSaware specifications and architecture - M24	M(ust)	TAU	The base functionality works.	D2.3	Achieved

	a single OpenCL runtime.										
TC4.1.1.NFR1	The implementation overhead of the wrapper should be less than 1% of the wrapped design.	Functionality & integration	DoA	WP4	TC4.1.1 OpenCL Wrapper for Hardware IP Cores	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	TAU	The base functionality works.	D2.3	Achieved
TC4.1.2.R1	HLS based SW to HW Transformation	Efficiency	End User & DoA	WP4	TC4.1.2 Profiling	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	HLS implementation of the DSM kernels	D4.1	Partially achieved
TC4.1.2.R2	Commissioning: The component should be able to collect hardware bitstreams IP Cores and download them on the FPGA fabric of a Multiprocessor System on Chip FPGA board.	Efficiency	End User & DoA	WP4	TC4.1.2 Profiling	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Vitis XRT is used to implement HW kernel commissioning in the DSM module	D4.1	Partially achieved
TC4.1.2.R3	Reconfigurability: The components should be able to reconfigure the commissioned hardware IP Cores on the FPGA fabric	Efficiency	End User & DoA	WP4	TC4.1.2 Profiling	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Vitis XRT is used to support dynamic HW reconfiguration	D4.1	Partially achieved
TC4.1.2.R4	Multiprocessor System on Chip FPGA board and replace existing hardware IP Cores.	Efficiency	End User & DoA	WP4	TC4.1.2 Profiling	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	FPGAs with multiple ARM processors were employed	D4.1	Partially achieved
TC4.1.2.R5	Removal: The component should	Efficiency	End User & DoA	WP4	TC4.1.2 Profiling	MS7 - Intermediate CPHs Architecture	M(ust)	UOP	Dynamic reconfiguration of HW	D4.1	Partially achieved

	be able to remove existing hardware IP Cores in the FPGA fabric of a Multiprocessor System on Chip (MPSoS) FPGA board.					Design and Implementation - M24			kernels implies HW kernel removal		
TC4.1.2.R6	Accessibility: The component should be able to communicate with the model based design mechanism of the CPSoSaware layer in order to deploy hardware IP Cores in the MPSoC board.	Efficiency	End User & DoA	WP4	TC4.1.2 Profiling	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Implemented as part of the dynamic reconfiguration of HW kernels	D4.1	Partially achieved
TC4.1.2.R7	IP Core Software Support: The component should be able to deploy appropriate software driver components on the runtime system (embedded OS or bare metal API) been executed on a MPSoC FPGA board so that hardware IP Cores are accessible. Support for POCL tool could be offered	Efficiency	End User & DoA	WP4	TC4.1.2 Profiling	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Xilinx XRT requires appropriate drivers to support the configuration of any HW kernel. PoCL integration is also investigated	D4.1	Partially achieved
TC4.1.2.R8	Accelerate DNN inference in comparison to	Efficiency	End User & DoA	WP4	TC4.1.2 Profiling	MS7 - Intermediate CPHs Architecture Design and	M(ust)	UOP	CNN for handwritten character recognition	D4.1	Partially achieved

	software running in ARM					Implementation - M24			implemented using PoCL interface		
TC4.1.2.NFR1	Reliability and robustness of the suggested assembly steps.	Reliability	End User & DoA	WP4	TC4.1.2 Profiling	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Reliability and robustness of assembly is guaranteed by Xilinx Vitis, XRT. Additional checks will be used where applicable	D4.1	Partially achieved
TC4.1.2.NFR2	Programmable co-processor for cases where hardware. (TC3.6.1.R1) customization is useful, but runtime programmability is needed.	Efficiency	End User & DoA	WP4	TC4.1.2 Profiling	MS6 - Preliminary CPHs Architecture Design and Implementation - M12	C(ould)	UOP	Will be investigated if applicable	D4.1	Not achieved yet
TC4.1.2.NFR3	Performance requirements are task/application specific. Overall, acceleration or improved energy-efficiency over similar software on a general-purpose processor is required to justify an ASIP.	Efficiency	End User & DoA	WP4	TC4.1.2 Profiling	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Comparison is performed concerning the accuracy, speed, energy consumption between accelerated functions and their original SW implementations (e.g. DSM module)	D4.1	Partially achieved
TC4.1.3.R1	Input. The component must handle input in a mathematical optimization format, providing the necessary	Functionality	End User & DoA	WP4	TC4.1.3 Architecture Optimization	MS8 - Final CPHs Architecture Design and Implementation - M36	M(ust)	IBM	Will be adapted from CERBERO[4]	D4.6	Not achieved yet

[4] https://www.cerbero-h2020.eu/

	abstractions to model (with decision variables) CPSs/CPHSs including both hardware and software components and their connections.										
TC4.1.3.R2	Objective. The component should be capable of optimizing a variety of objective functions. This does not include simultaneous multiple objectives (Pareto front).	Functionality	End User & DoA	WP4	TC4.1.3 Architecture Optimization	MS8 - Final CPHs Architecture Design and Implementation - M36	S(hould)	IBM	Will be adapted from CERBERO	D4.6	Not achieved yet
TC4.1.3.R3	Constraints. The component must be able to handle connection, application, and resource constraints.	Functionality	End User & DoA	WP4	TC4.1.3 Architecture Optimization	MS8 - Final CPHs Architecture Design and Implementation - M36	M(ust)	IBM	Will be adapted from CERBERO	D4.6	Not achieved yet
TC4.1.3.R4	Output. The component should produce as output a hardware-software partitioning that is optimal according to the specified mathematical optimization problem.	Functionality	End User & DoA	WP4	TC4.1.3 Architecture Optimization	MS8 - Final CPHs Architecture Design and Implementation - M36	S(hould)	IBM	Will be adapted from CERBERO	D4.6	Not achieved yet

TC4.1.3.NFR1	Efficiency (response time)	Efficiency	End User & DoA	WP4	TC4.1.3 Architecture Optimization	MS8 - Final CPHs Architecture Design and Implementation - M36	C(ould)	IBM	Will be adapted from CERBERO	D4.6	Not achieved yet
TC4.1.3.NFR2	Efficiency (optimality)	Efficiency	End User & DoA	WP4	TC4.1.3 Architecture Optimization	MS8 - Final CPHs Architecture Design and Implementation - M36	C(ould)	IBM	Will be adapted from CERBERO	D4.6	Not achieved yet
TC4.1.3.NFR3	Feasibility of solution	Functionality	End User & DoA	WP4	TC4.1.3 Architecture Optimization	MS8 - Final CPHs Architecture Design and Implementation - M36	S(hould)	IBM	Will be adapted from CERBERO	D4.6	Not achieved yet
TC4.2.1.R1	SW agents running on the HW platform monitor the network performance under the current network configuration for specific application scenario	Reliability	End User & DoA	WP4	TC4.2.1 Intra-Communication Manager	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Metrics of the network traffic will be recorded in application level	D3.2 & D5.2	Not achieved yet
TC4.2.1.R2	The performance outcome is processed in order to evaluate whether the application requirements are met	Efficiency	DoA	WP4	TC4.2.1 Intra-Communication Manager	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	UOP	Metrics of the network traffic will be recorded in application level	D3.2 & D5.2	Not achieved yet
TC4.2.1.R3	SW agent running on the HW is responsible to receive new network configuration and/or network interface firmware	Functionality & Integration	End User & DoA	WP4	TC4.2.1 Intra-Communication Manager	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	MQTT subscriber listen for configuration updates on the network properties	D3.2 & D5.2	Achieved

	to apply on the device										
TC4.2.1.NFR1	The device should be able to recover from failing network firmware/configuration update	Reliability	End User & DoA	WP4	TC4.2.1 Intra-Communication Manager	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	UOP	Handle exceptional cases and revert to previous state	D3.2 & D5.2	Partially achieved
TC4.2.1.NFR2	SW agent should be able to verify the integrity of the received payloads	Reliability	DoA	WP4	TC4.2.1 Intra-Communication Manager	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Hashing algorithms will be used to verify that received payloads has been successfully received	D3.2 & D5.2	Achieved
TC4.2.1.NFR3	Versioning of the applied configurations should be supported	Maintainability	DoA	WP4	TC4.2.1 Intra-Communication Manager	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	UOP	A version property will be part of the commissioning payload	D3.2 & D5.2	Achieved
TC4.2.1.NFR4	Authentication/Authorization for receiving configuration updates	Security	End User & DoA	WP4	TC4.2.1 Intra-Communication Manager	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Two authentication/authorization schemes will be investigated. 1. Basic authentication 2. Mutual SSL	D3.2 & D5.2	Not achieved yet
TC4.3.1.R1	Input. The component must receive normalized security events through TCP/41000 from agents/sensors deployed remotely, in the infrastructure that is under surveillance. Events comply with	Functionality & integration	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	ATOS	ATOS provides sensors and integrates them with the SRMM. If some partner provides sensors from their side, ATOS can integrate them mainly by normalizing the data format	D3.5, D4.3, D4.8	Achieved

	a predefined JSON format.										
TC4.3.1.R2	Configuration. The component should be configured using the component's graphical dashboard, to define the security monitoring infrastructure in use (topology of sensors/agents deployed and active), the security detection rules and the correlation directives.	Functionality & integration	End User & DoA	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	ATOS	Two possible ways: either using the existing interface developed by ATOS from other projects, and making needed adaptations, or developing a specific configuration interface in CPSoSAware. This remains to be decided	D2.2, D4.3, D4.8	Not achieved yet
TC4.3.1.R3	Events Processing. The component must process security events received as input, correlate them using the security detection rules configured, and generate security alarms as output, as defined in the correlation directives configured.	Functionality	End User & DoA	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	ATOS	We need to define the needed correlation rules to be applied in the SRMM	D4.3, D4.8	Partially achieved
TC4.3.1.R4	Output. The component should produce as output security alarms. Alarms comply	Functionality & integration	End User & DoA	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	ATOS	Alarms are produced following the internal correlation process of the SRMM. JSON format to be confirmed	D4.3, D4.8	Partially achieved

	with a predefined JSON format. Alarms can be configured to be persisted in a DB, logged into a file, transmitted to a third-party component (using a middleware such as Message Queue/Broker) and displayed in the SRMM graphical dashboard.								within the Consortium as it will have to be used by the CSAIE (T2.1). Message brokering technology needs to be confirmed. We have preference for AMQP or Kafka		
TC4.3.1.R5	Cross-correlation. Security alarms produced as output by the SRMM can be configured to be input into the SRMM correlation engine, for cross-correlation processes.	Functionality	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS8 - Final CPHs Architecture Design and Implementation - M36	C(ould)	ATOS	The capability of performing cross-correlation already exists. It is to be expected that new rules are produced in the context of the project	D4.3, D4.8	Partially achieved
TC4.3.1.NFR1	Scalability - of the SRMM correlation engine and data collection module	Maintainability	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS8 - Final CPHs Architecture Design and Implementation - M36	M(ust)	ATOS	It will be achieved by enhancements made to the assets during the project	D4.8	Partially achieved
TC4.3.1.NFR2	High-performance - of the SRMM correlation engine and the data persistence layer	Efficiency	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS8 - Final CPHs Architecture Design and Implementation - M36	M(ust)	ATOS	During the project we will research on how to improve the performance of the asset, which is currently high. We still have no information on how stressed the component will be during the piloting tests	D4.8, D6.3	Partially achieved

TC4.3.1.NFR3	Integrity - of the security events transmitted from sensors/agents to the SRMM component, and of the security alarms generated as output by the SRMM	Security	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	ATOS	Already achieved	D4.3	Achieved
TC4.3.1.NFR4	Confidentiality - of the security events transmitted from sensors/agents to the SRMM component, and of the security alarms generated as output by the SRMM	Security	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	ATOS	Already achieved	D4.3	Achieved
TC4.3.1.NFR5	Accountability - of the security events transmitted from sensors/agents to the SRMM component, of the correlation process and of the security alarms generated as output by the SRMM	Security	End user	WP4	TC4.3.1 Security Runtime Monitoring	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	ATOS	Already achieved	D4.3	Achieved
TC4.4.1.R1	ROS/ROS2 interface	Functionality & integration	End user	WP4	TC4.4.1 V2X simulator		C(ould)	I2CAT			Not achieved yet
TC4.4.1.R2	V2X representation of state in AV simulator	Functionality & integration	End User & DoA	WP4	TC4.4.1 V2X simulator	MS10 - Final version of the Simulation Software - M24	M(ust)	I2CAT		D4.2	Partially achieved

TC4.4.1.NFR1	Possibility of running in real time	Efficiency	End user	WP4	TC4.4.1 V2X simulator		W(on't)	I2CAT	The simulation of V2X message transmission takes a lot of time and it is not feasible to run it in real time as the AV simulator does.		Not achieved yet
TC4.4.1.NFR2	Modular architecture integrated in simulation framework	Maintainability	End user	WP4	TC4.4.1 V2X simulator	MS10 - Final version of the Simulation Software - M24	M(ust)	I2CAT	Already achieved	D.4.2	Achieved
TC4.4.3.R1	Machine learning support for perception algorithms	Functionality	DoA	WP4	TC4.4.3 AV Simulation	MS10 - Final version of the Simulation Software - M24	M(ust)	RTC	In terms of support for Machine Learning, sensor data together with corresponding labels are generated by the simulator.	D2.2	Partially achieved
TC4.4.3.R2	User control: Simulation should allow users to control all critical aspects in the simulation through dedicated API (e.g. agents behavior or sensors).	Functionality	DoA	WP4	TC4.4.3 AV Simulation	MS10 - Final version of the Simulation Software - M24	M(ust)	RTC	Simulation can be controlled by json configuration files and Python API	D2.2	Partially achieved
TC4.4.3.R3	Integration with middleware: Simulation solution should offer integration with state-of-the-art robotics middleware (e.g. ROS and ROS2)	Integration	DoA	WP4	TC4.4.3 AV Simulation	MS10 - Final version of the Simulation Software - M24	M(ust)	RTC	Simulator is integrated with ROS2 middleware	D2.2	Achieved
TC4.4.3.NFR1	Simple way of defining test scenarios	Functionality	DoA	WP4	TC4.4.3 AV Simulation	MS10 - Final version of the Simulation Software - M24	M(ust)	RTC	Definition of test scenarios in Python API and json files	D2.2	Achieved

TC4.4.3.NFR2	Scalability to multiple agents control - Simulation should provide multiple clients that can control different actors.	Functionality	DoA	WP4	TC4.4.3 AV Simulation	MS10 - Final version of the Simulation Software - M24	M(ust)	RTC	Multiple agents support implemented	D2.2	Achieved
TC4.4.3.NFR3	Scalability to cloud services - Simulation should be able to run on scalable cloud services to run multiple simulation scenarios (e.g. Google Cloud, Microsoft Azure or other).	Integration	DoA	WP4	TC4.4.3 AV Simulation	MS10 - Final version of the Simulation Software - M24	S(hould)	RTC	Cloud deployment of the simulator will be implemented	D2.2	Not achieved yet
TC4.4.3.NFR4	Fast execution: Software should offer a fast execution simulation for which graphics are not required.	Functionality	DoA	WP4	TC4.4.3 AV Simulation	MS10 - Final version of the Simulation Software - M24	S(hould)	RTC	The simulator is highly optimized for fast and real-time execution of test scenarios	D2.2	Partially achieved
TC4.4.3.NFR5	Diagnostic and Error Handling - Simulation should offer diagnostic and error handling	Functionality	DoA	WP4	TC4.4.3 AV Simulation	MS10 - Final version of the Simulation Software - M24	S(hould)	RTC	Diagnostics will be implemented, currently only initialized.	D2.2	Partially achieved
TC4.4.3.NFR6	Determinism - Simulation should ensure determinism	Reliability	DoA	WP4	TC4.4.3 AV Simulation	MS10 - Final version of the Simulation Software - M24	S(hould)	RTC	Simulation is mostly deterministic	D2.2	Partially achieved
TC4.4.3.NFR7	Modular System Architecture - Simulation should	Functionality & integration	DoA	WP4	TC4.4.3 AV Simulation	MS10 - Final version of the Simulation Software - M24	M(ust)	RTC	Architecture of all simulation components is modular	D2.2	Achieved

	have modular system architecture										
TC4.5.1.R1	Ontology schemas should be expressed in RDF, n-triples, OWL or other established ontology formats	Functionality	End User & DoA	WP4	TC4.5.1 Semantic Knowledge Graph	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	CTL	Ontology schemas have been developed using established editors like Protege. The exported files are expressed in RDF-compliant formats, such as Turtle (.ttl) and n-triples (.nt).	D4.5	Achieved
TC4.5.1.R2	The deployed RDF triplestore should provide a SPARQL-enabled endpoint (API).	Functionality & integration	End User & DoA	WP4	TC4.5.1 Semantic Knowledge Graph	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	CTL	The free version of GraphDB has been selected as triplestore. It provides a SPARQL endpoint.	D4.5	Achieved
TC4.5.1.R3	The RDF triplestore should support SHACL.	Functionality	End User & DoA	WP4	TC4.5.1 Semantic Knowledge Graph	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	C(ould)	CTL	GraphDB supports SHACL, however this feature will not be used in TC4.5.1.	D4.5	Redundant
TC4.5.1.R4	The RDF triplestore should support concurrent execution of queries	Functionality	End User & DoA	WP4	TC4.5.1 Semantic Knowledge Graph	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	CTL	The free version of GraphDB supports up to two concurrent queries. This is adequate, as concurrency (when implemented) can be easily extended to more than two queries.	D4.5	Achieved
TC4.5.1.NFR1	Domain experts should support the definition of the ontology schema by providing domain knowledge to the semantic experts.	Reliability	End User & DoA	WP4	TC4.5.1 Semantic Knowledge Graph	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	CTL	Collaboration with CPSoSaware partners (domain experts and component owners)	D4.5	Achieved

TC4.5.1.NFR2	The RDF tiplestore should be on industry level, able to handle several millions of RDF triples.	Efficiency	End User & DoA	WP4	TC4.5.1 Semantic Knowledge Graph	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	CTL	GraphDB is industry level	D4.5	Achieved
TC4.5.2.R1	The component should support concurrent requests.	Functionality & integration	End User & DoA	WP4	TC4.5.2 Semantic Knowledge Graph Service	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	CTL	The service is implemented as a REST API, which supports concurrent request.	D4.5	Achieved
TC4.5.2.R2	The component should provide services for data population to TC4.5.1	Functionality & integration	End User & DoA	WP4	TC4.5.2 Semantic Knowledge Graph Service	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	CTL	SPARQL queries to TC4.5.1 endpoint.	D4.5	Achieved
TC4.5.2.R3	The component should provide services for data retrieval from TC4.5.1	Functionality & integration	End User & DoA	WP4	TC4.5.2 Semantic Knowledge Graph Service	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	CTL	SPARQL queries to TC4.5.1 endpoint.	D4.5	Achieved
TC4.5.2.R4	The component should allow the semantic reasoning mechanism to be triggered by other component requests.	Functionality & integration	End User & DoA	WP4	TC4.5.2 Semantic Knowledge Graph Service	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	CTL	Appropriate API services will be implemented to trigger reasoning.	D4.5	Partially achieved
TC4.5.2.R5	The component should be able to apply different reasoning rulesets in a modular way.	Functionality	End User & DoA	WP4	TC4.5.2 Semantic Knowledge Graph Service	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	CTL	Reasoning rulesets will be defined as SPARQL queries stored in JSON-formatted files.	D4.5	Not achieved yet
TC4.5.2.NFR1	Domain experts and end-users should support the definition of meaningful	Reliability	End User & DoA	WP4	TC4.5.2 Semantic Knowledge Graph Service	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	CTL	Collaboration with CPSoSaware partners (domain experts and component owners)	D4.5	Partially achieved

	semantic reasoning rules.										
TC4.5.2.NFR2	Other component owners should express requirements for specific API services for data insertion/retrieval.	Integration	End User & DoA	WP4	TC4.5.2 Semantic Knowledge Graph Service	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	CTL	Collaboration with CPSoSaware partners (domain experts and component owners)	D4.5	Partially achieved
TC4.5.2.NFR3	The provided services should be as generic/re-usable as possible, with multiple parameters for result customization.	Usability	End User & DoA	WP4	TC4.5.2 Semantic Knowledge Graph Service	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	CTL	REST API design and implementation considers this requirement.	D4.5	Partially achieved
TC4.6.1.R1	Commissioning: The component should be able to collect hardware bitstreams IP Cores and download them on the FPGA fabric of a Multiprocessor System on Chip FPGA board			WP4	TC4.6.1 Commissioning of Hardware Components in CPSs			IBM			Obsolete
TC4.6.1.R2	Reconfigurability: The components should be able to reconfigure the commisioned hardware IP Cores on the FPGA fabric			WP4	TC4.6.1 Commissioning of Hardware Components in CPSs			IBM			Obsolete
TC4.6.1.R3	Multiprocessor System on Chip FPGA board and			WP4	TC4.6.1 Commissioning of Hardware			IBM			Obsolete

	replace existing hardware IP Cores				Components in CPSs						
TC4.6.1.R4	Removal: The component should be able to remove existing hardware IP Cores in the FPGA fabric of a Multiprocessor System on Chip (MPSoS) FPGA board			WP4	TC4.6.1 Commissioning of Hardware Components in CPSs			IBM			Obsolete
TC4.6.1.R5	Accesibility: The component should be able to communicate with the model based design mechanism of the CPSoSaware layer in order to deploy hardware IP Cores in the MPSoC board			WP4	TC4.6.1 Commissioning of Hardware Components in CPSs			IBM			Obsolete
TC4.6.1.R6	IP Core Software Support: The component should be able to deploy appropriate software driver components on the runtime system (embedded OS or bare metal API) been executed on a MPSoC FPGA board so that hardware IP Cores are accessible. Support			WP4	TC4.6.1 Commissioning of Hardware Components in CPSs			IBM			Obsolete

	for POCL tool could be offered										
TC4.6.1.NFR1	The component should be able to validate that connectivity exists and recover from possible network failures.			WP4	TC4.6.1 Commissioning of Hardware Components in CPSs			IBM			Obsolete
TC4.6.1.NFR2	The component should be able to handle efficiently the configuration updates and resolve any possible dependencies.			WP4	TC4.6.1 Commissioning of Hardware Components in CPSs			IBM			Obsolete
TC4.6.1.NFR3	The component should be able to provide integrity validation method in both ends (e.g. hashes of the transferred payloads).			WP4	TC4.6.1 Commissioning of Hardware Components in CPSs			IBM			Obsolete
TC4.6.1.NFR4	The component should be aware of the commissioning process' status and handle failures (e.g. rollback to previous versions).			WP4	TC4.6.1 Commissioning of Hardware Components in CPSs			IBM			Obsolete
TC5.1.1.R1	Profiling	Efficiency	End User & DoA	WP5	TC5.1.1 HLS based SW to HW Transformation	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Profiling using Xilinx Vitis	D5.1	Partially achieved

TC5.1.1.R2	Commissioning of Hardware Components in CPSs	Functionality	End User & DoA	WP5	TC5.1.1 HLS based SW to HW Transformation	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Implemented using Xilinx XRT	D5.1	Partially achieved
TC5.1.1.R3	Reconfigurability	Efficiency	End User & DoA	WP5	TC5.1.1 HLS based SW to HW Transformation	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	XRT will be used to implement dynamic reconfiguration of kernels and face alignment model switching	D5.1	Partially achieved
TC5.1.1.R4	IP Core Software Support	Functionality	End User & DoA	WP5	TC5.1.1 HLS based SW to HW Transformation	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	HW kernels and their drivers are developed and tested simultaneously in Xilinx Vitis	D5.1	Partially achieved
TC5.1.1.R5	ML Hardware Accelerator IP Cores	Functionality	End User & DoA	WP5	TC5.1.1 HLS based SW to HW Transformation	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Face alignment and CNN for handwritten character recognition have been implemented on FPGA	D5.1	Partially achieved
TC5.1.1.R6	Accelerate DNN inference in comparison to software running in ARM.	Efficiency	End User & DoA	WP5	TC5.1.1 HLS based SW to HW Transformation	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	CNN for handwritten character recognition implemented both in SW and HW and compared	D5.1	Partially achieved
TC5.1.1.R7	Provide access to all OpenCL-supported devices in a network distributed platform from a single host application.	Functionality	End User & DoA	WP5	TC5.1.1 HLS based SW to HW Transformation	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	CNN for handwritten character recognition implemented using PoCL	D5.1	Partially achieved
TC5.1.1.NFR1	Development of HW-SW Library with reliable Components.	Efficiency	End User & DoA	WP5	TC5.1.1 HLS based SW to HW Transformation	MS7 - Intermediate CPHs Architecture Design and	M(ust)	UOP	FPGA implementations of DSM components as well as other HSL	D3.6	Partially achieved

						Implementation - M24			components used to populate this library		
TC5.1.1.NFR2	Performance requirements are task/application specific. Overall, acceleration or improved energy-efficiency over similar software on a general-purpose processor is required to justify an ASIC.	Efficiency	End User & DoA	WP5	TC5.1.1 HLS based SW to HW Transformation	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	M(ust)	UOP	Comparison is performed concerning the accuracy, speed, energy consumption between accelerated functions and their original SW implementations (e.g. DSM module)	D4.1	Partially achieved
TC5.3.1.R1	Involve gamification of learning which makes the process fun and interactive.	Usability	DoA	WP5	TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	MS8 - Final CPHs Architecture Design and Implementation - M36	S(hould)	UPAT	Development of a XR learning tool/interface for getting trained in VR, supported by visual hints and feedback on performance,	D5.2	Not achieved yet
TC5.3.1.R2	Provide visual cues in a distraction-free environment which helps the users to better understand the concepts.	Efficiency	DoA	WP5	TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	MS8 - Final CPHs Architecture Design and Implementation - M36	M(ust)	UPAT	Visualization of safety zones in AR	D5.2	Partially achieved
TC5.3.1.R3	The technologies come with intelligent learning content and provide real-time responses.		DoA	WP5	TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS		W(on't)	UPAT		D5.2	Rejected
TC5.3.1.R4	The trainee can easily accomplish the mapping	Functionality	DoA	WP5	TC5.3.1 Extended Reality lifelong	MS7 - Intermediate CPHs Architecture Design and	S(hould)	UPAT	Development of new training material based on interactive	D5.2	Not achieved yet

	between the training and the real task and is also able to access additional training material or information about the virtual objects.				learning tools/Interfaces for integrated CPSoS	Implementation - M24			simulations of the work task		
TC5.3.1.R5	AR/VR can support assembly tasks in hybrid human-machine manufacturing lines, improving efficiency and ergonomics.	Efficiency	DoA	WP5	TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	MS8 - Final CPHs Architecture Design and Implementation - M36	C(ould)	UPAT	Training scenario without the need of an actual human user.	D5.2	Not achieved yet
TC5.3.1.NFR1	Computational efficiency (real-time).	Efficiency	DoA	WP5	TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	MS8 - Final CPHs Architecture Design and Implementation - M36	S(hould)	UPAT	Parameterization of the algorithm to be computationally efficient	D5.2	Not achieved yet
TC5.3.1.NFR2	User-friendly interface.	Usability	DoA	WP5	TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	MS3 - CPSoSaware Final architecture - M36	S(hould)	UPAT	The provided information will be easy to be understood by all users	D5.2	Not achieved yet
TC5.3.1.NFR3	Reliability and robustness of the suggested assembly steps.	Reliability	DoA	WP5	TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	UPAT	The suggested assembly steps will be based on the personalized users' preference and experience	D5.2	Partially achieved

TC5.3.1.NFR4	Improve the learning procedure without disturbing the user's attention.	Usability	DoA	WP5	TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	UPAT	Providing intuitive and non-distractive information so that to simplify the training process and emphasize the learning procedure	D5.2	Partially achieved
TC5.3.1.NFR5	Provide only these type of help and instructions based on personalized user's preferences.	Usability	DoA	WP5	TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	MS7 - Intermediate CPHs Architecture Design and Implementation - M24	S(hould)	UPAT	Taking into account the personal preferences of the users during the training process into the simulators	D5.2	Partially achieved
TC5.3.2.R1	Machine learning support			WP5	TC5.3.2 Manufacturing Environment Simulation			UPAT			Obsolete
TC5.3.2.R2	Possibility of modelling additional elements of use case scenarios: humans, light curtain, safety eye, etc.	Functionality	DoA	WP5	TC5.3.2 Manufacturing Environment Simulation	MS3 - CPSoSaware Final architecture - M36	S(hould)	UPAT	Already achieved	D5.3	Achieved
TC5.3.2.R3	Available models of robotic arms used in CRF factory	Functionality	DoA	WP5	TC5.3.2 Manufacturing Environment Simulation	MS3 - CPSoSaware Final architecture - M36	S(hould)	UPAT	Designing digital twins of the robotic models that are used in the CRF factory.	D5.3	Not achieved yet
TC5.3.2.R4	Integration with middleware: Simulation solution should offer integration with state-of-the-art robotics			WP5	TC5.3.2 Manufacturing Environment Simulation			UPAT			Obsolete

	middleware (e.g. ROS and ROS2)										
TC5.3.2.R5	User control: Simulation should allow users to control all critical aspects in the simulation through dedicated API (e.g. agents behavior or sensors).			WP5	TC5.3.2 Manufacturing Environment Simulation			UPAT			Obsolete
TC5.3.2.NFR1	Fast execution - Software should offer a fast execution simulation for which graphics are not required.	Efficiency	DoA	WP5	TC5.3.2 Manufacturing Environment Simulation	MS3 - CPSoSaware Final architecture - M36	C(ould)	UPAT	Run in real time.	D5.3	Partially achieved
TC5.3.2.NFR2	Diagnostic and Error Handling - Simulation should offer diagnostic and error handling	Reliability	DoA	WP5	TC5.3.2 Manufacturing Environment Simulation	MS3 - CPSoSaware Final architecture - M36	C(ould)	UPAT	Designing simulation scenarios in which real world errors could happen.	D5.3	Partially achieved
TC5.3.2.NFR3	Determinism - Simulation should ensure determinism	Reliability	DoA	WP5	TC5.3.2 Manufacturing Environment Simulation	MS3 - CPSoSaware Final architecture - M36	C(ould)	UPAT	Creating realistic scenarios.	D5.3	Partially achieved
TC5.3.2.NFR4	Modular System Architecture - Simulation should have modular system architecture			WP5	TC5.3.2 Manufacturing Environment Simulation			UPAT			Obsolete

A total of 232 requirements have been recorded, presenting the following characteristics (Figures 8 to 10).

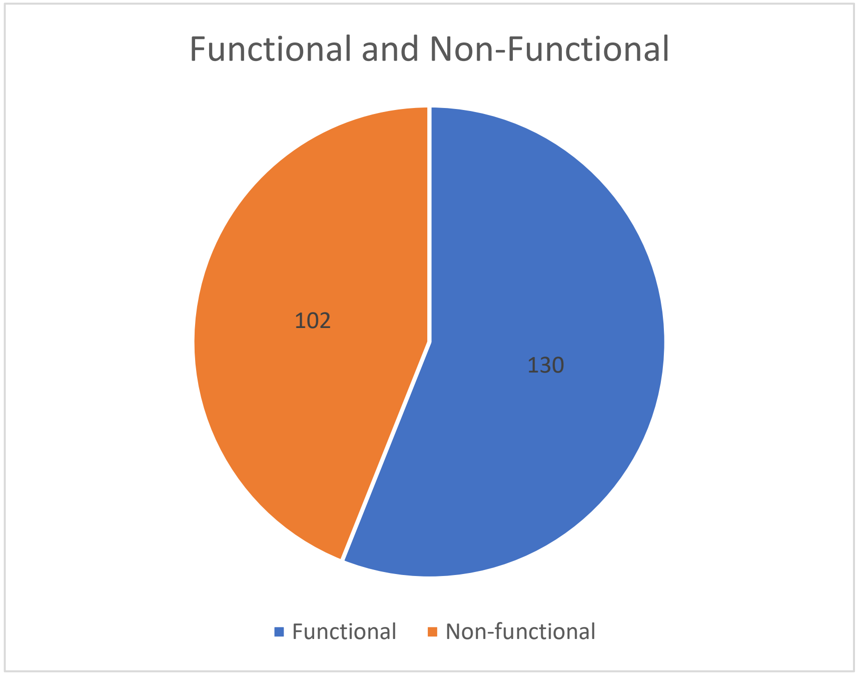

Figure 8 - Functional and non-functional requirement distribution

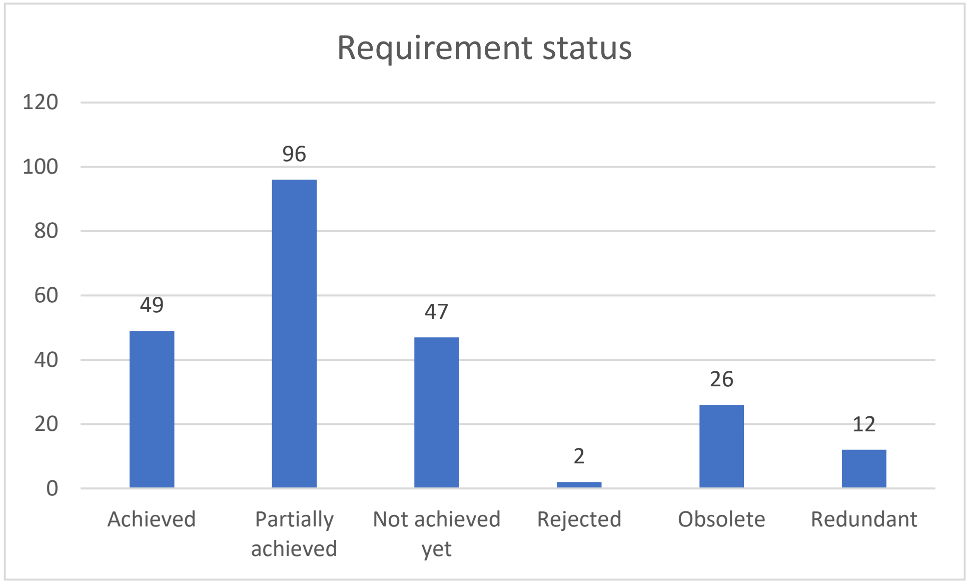

Figure 9 - Fulfilment status of requirements

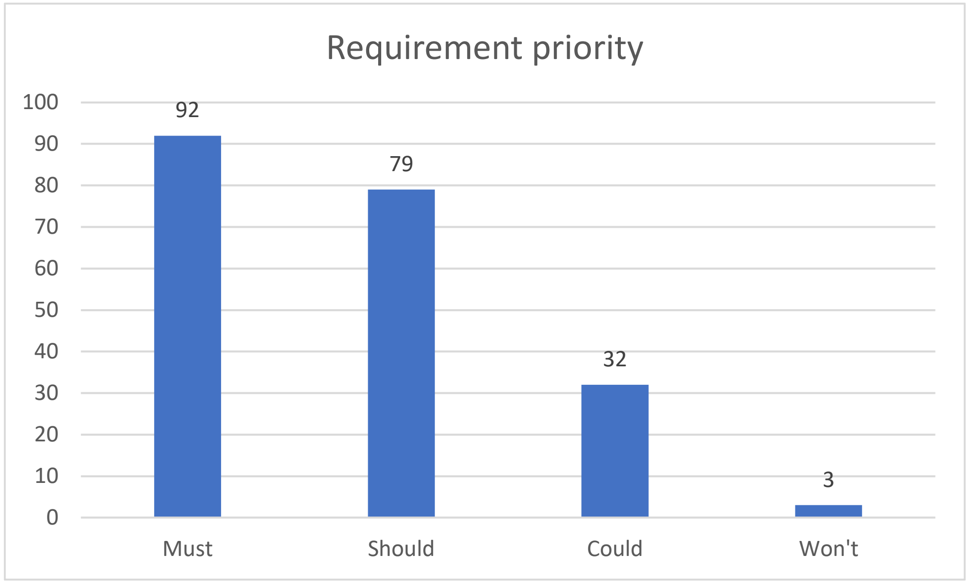

Figure 10 - Requirement priority distribution

5 Distribution View

This section demonstrates the logical distribution of components within the system architecture.

5.1 System components per Use Case

Technical Component	UC1 – Connected and Autonomous Vehicles	UC2 - Human-Robot Interaction in Manufacturing Environment
TC2.2.1 Intra-Communication Sim Tool	X	X
TC2.2.2 pocl-remote	X	
TC2.3.1 ML Hardware Accelerator IP Cores	X	X
TC2.3.2 Security Accelerators for CPS security agents/sensors	X	
TC2.3.3 Model transformation to openCL	X	
TC2.4.1 Xilinx XRT KPI monitoring	X	X
TC2.5.1 Modelling Orchestration Tool	X	
TC3.1.1 Visual Localization	X	
TC3.1.2 Deep Multimodal Scene Understanding	X	
TC3.1.3 User Behaviour Monitoring	X	X
TC3.1.4 AI Acceleration	X	
TC3.2.1 pocl-accel	X	
TC3.3.1 Multimodal Localization API	X	
TC3.3.2 PathPlanning API	X	
TC3.4.1 XR tools for increasing situational awareness	X	X
TC3.5.1 CPS layer Security sensors/agents	X	
TC3.6.1 TCE (openasip.org) soft cores	X	
TC4.1.1 OpenCL Wrapper for Hardware IP Cores	X	X
TC4.1.2 Profiling	X	X
TC4.1.3 Optimization	X	X
TC4.2.1 Intra-Communication Manager	X	X
TC4.3.1 Security Runtime Monitoring	X	

TC4.4.1 V2X simulator	X	
TC4.4.3 AV Simulation	X	
TC4.5.1 Semantic Knowledge Graph	X	X
TC4.5.2 Semantic Knowledge Graph Service	X	X
TC5.1.1 HLS based SW to HW Transformation	X	X
TC5.3.1 Extended Reality lifelong learning tools/Interfaces for integrated CPSoS	X	X
TC5.3.2 Manufacturing Environment Simulation		X

5.2 Architectural layers

As described in D1.3, the architectural perspective of layers in CPSoSaware consists of the following main blocks.

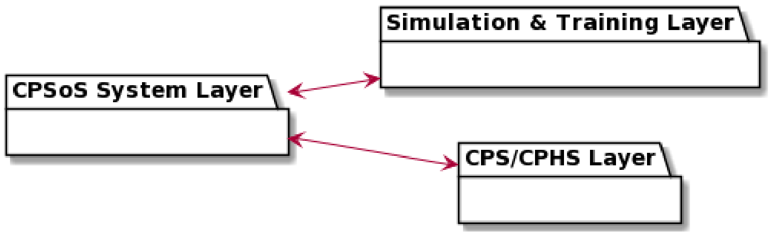

Figure 11 - Main architectural blocks

The updated distribution of technical components to these architectural blocks (and appropriate sub-blocks) is presented in Figures 12 to 14.

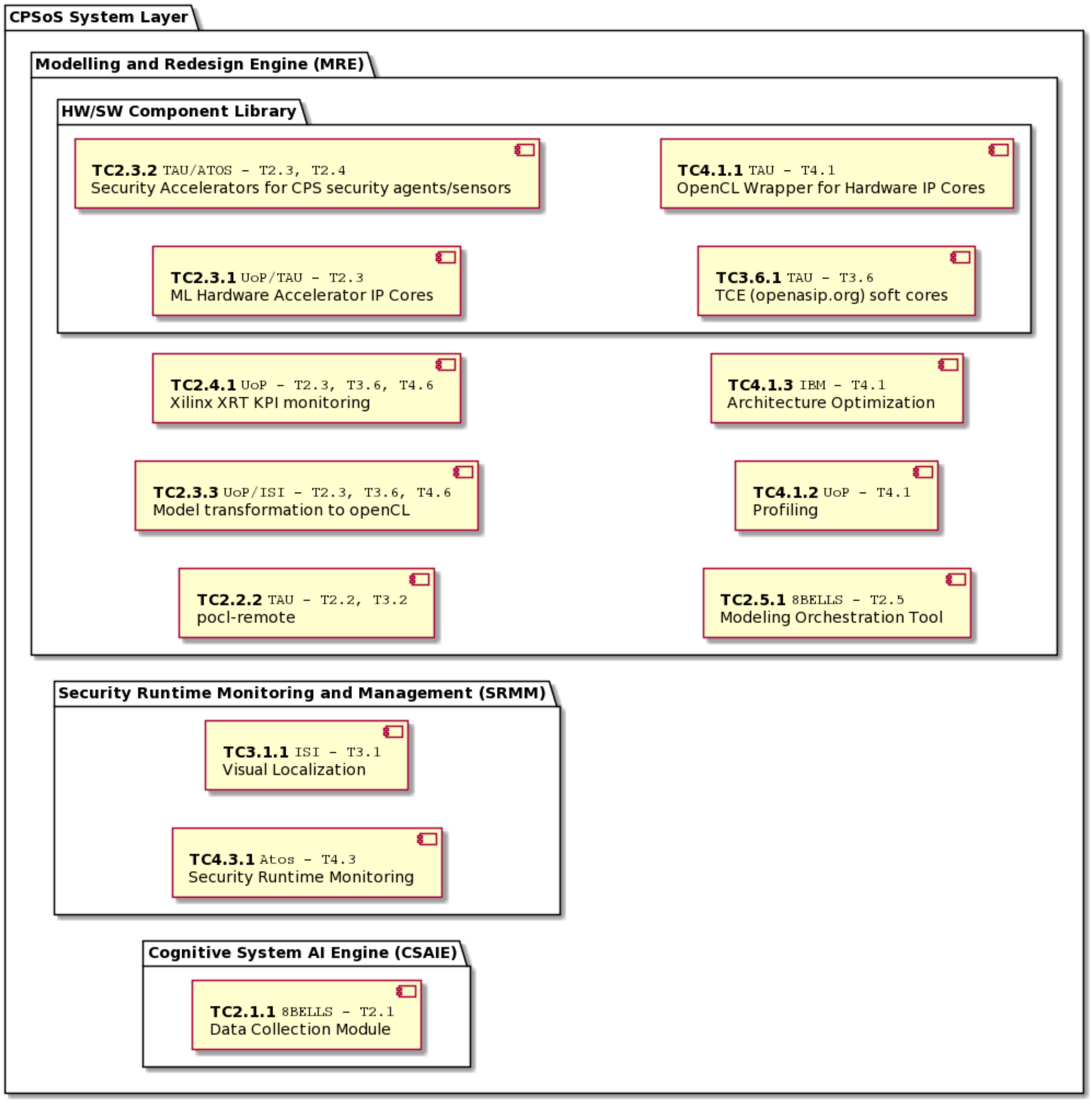
CPSoS System Layer
Modelling and Redesign Engine (MRE)
HW/SW Component Library
TC2.3.2 TAU/ATOS – T2.3, T2.4
Security Accelerators for CPS security agents/sensors
TC4.1.1 TAU – T4.1
OpenCL Wrapper for Hardware IP Cores
TC2.3.1 UoP/TAU – T2.3
ML Hardware Accelerator IP Cores
TC3.6.1 TAU – T3.6
TCE (openasip.org) soft cores
TC2.4.1 UoP – T2.3, T3.6, T4.6
Xilinx XRT KPI monitoring
TC4.1.3 IBM – T4.1
Architecture Optimization
TC2.3.3 UoP/ISI – T2.3, T3.6, T4.6
Model transformation to openCL
TC4.1.2 UoP – T4.1
Profiling
TC2.2.2 TAU – T2.2, T3.2
pocl-remote
TC2.5.1 8BELLS – T2.5
Modeling Orchestration Tool
Security Runtime Monitoring and Management (SRMM)
TC3.1.1 ISI – T3.1
Visual Localization
TC4.3.1 Atos – T4.3
Security Runtime Monitoring
Cognitive System AI Engine (CSAIE)
TC2.1.1 8BELLS – T2.1
Data Collection Module

Figure 12 - CPSoS layer and sub-blocks

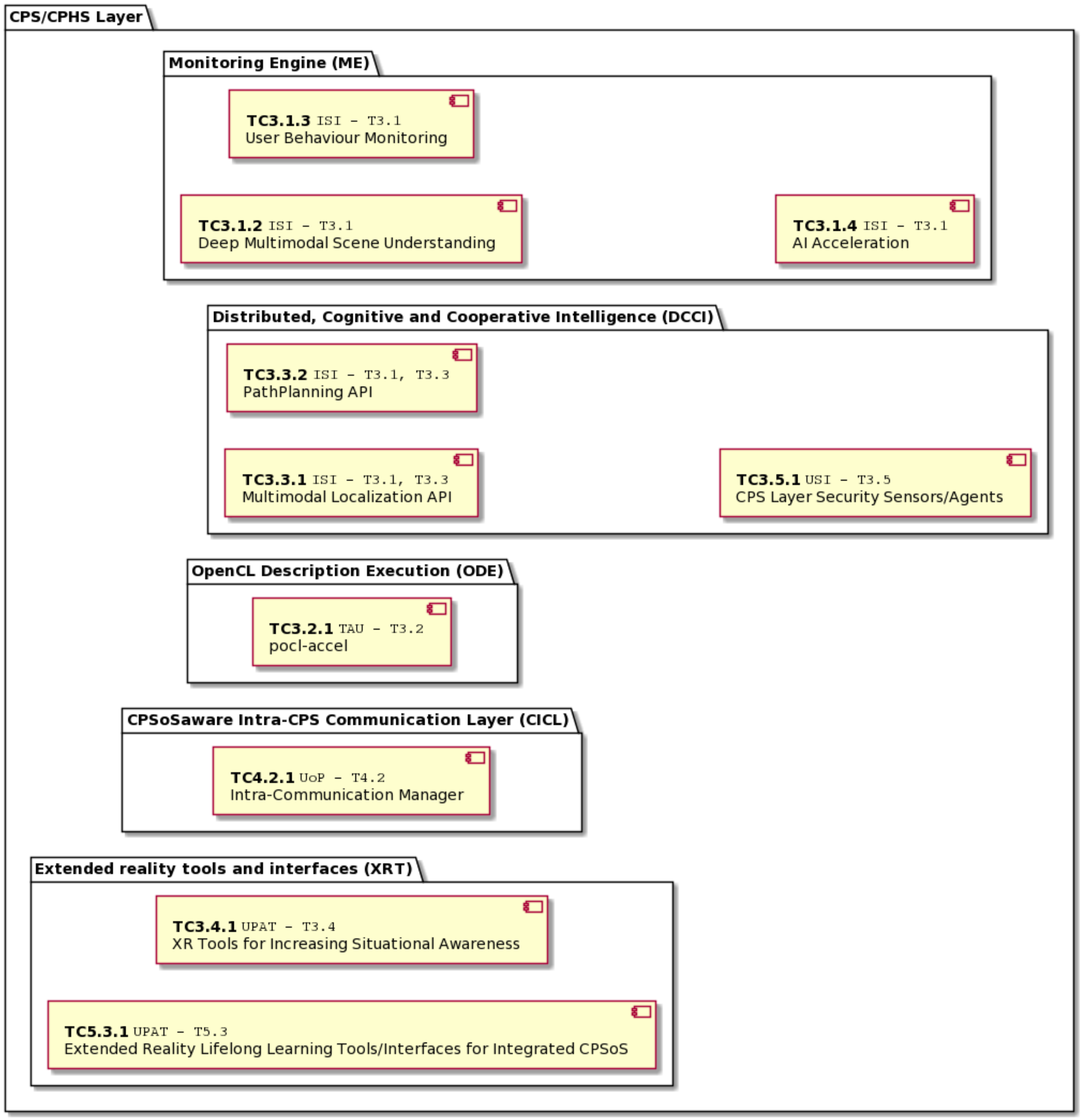
CPS/CPHS Layer
Monitoring Engine (ME)
TC3.1.3 ISI - T3.1 User Behaviour Monitoring
TC3.1.2 ISI - T3.1 Deep Multimodal Scene Understanding
TC3.1.4 ISI - T3.1 AI Acceleration
Distributed, Cognitive and Cooperative Intelligence (DCCI)
TC3.3.2 ISI - T3.1, T3.3 PathPlanning API
TC3.3.1 ISI - T3.1, T3.3 Multimodal Localization API
TC3.5.1 USI - T3.5 CPS Layer Security Sensors/Agents
OpenCL Description Execution (ODE)
TC3.2.1 TAU - T3.2 pocl-accel
CPSoSaware Intra-CPS Communication Layer (CICL)
TC4.2.1 UoP - T4.2 Intra-Communication Manager
Extended reality tools and interfaces (XRT)
TC3.4.1 UPAT - T3.4 XR Tools for Increasing Situational Awareness
TC5.3.1 UPAT - T5.3 Extended Reality Lifelong Learning Tools/Interfaces for Integrated CPSoS

Figure 13 - CPS/CPHS layer and sub-blocks

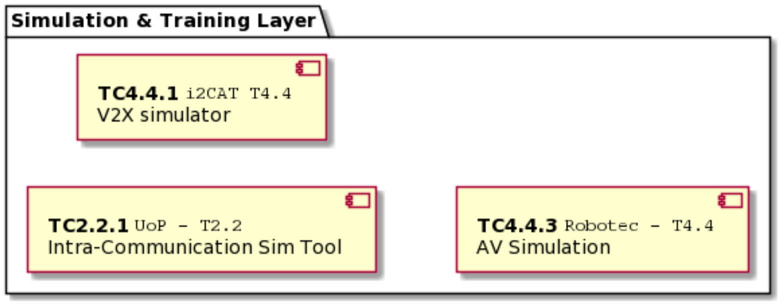

Figure 14 - Simulation and Training layer and sub-blocks

6 Conclusions and Next Steps

This deliverable initially introduced the applied ARCADE platform specification methodology and the related views. Building upon the progress described in D1.3 and documenting new inputs from all technical partners, this report presents the latest status of system components, focusing on deployment and interfacing requirements, and established dependencies. Subsequently, the document presents the updated list of technical requirements, which has been augmented with new fields that record important requirement aspects, such as the fulfilment status, target phase and priority. Finally, the distribution view is demonstrated, incorporating the administering of components in the CPSoSaware use cases and architectural blocks. Overall, this effort is expected to facilitate the implementation and integration of modules in the CPSoSaware system, acting as a guide and reference document for involved technical partners.

Monitoring the status of system requirements and interfaces between technical components periodically is a crucial task in order to achieve the designed system architecture. In this context, the revised reference document, along with the outcomes of the realisation view (see subsection 2.1), will be reported in the final version of this deliverable (D1.5, M36).

References

[1] IEEE, “Recommended Practice for Architectural Description for Software-Intensive Systems,” 2000.

[2] J. A. Khan, I. U. Rehman, Y. H. Khan, . I. J. Khan and S. Rashid, “Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique.,” *International Journal of Modern Education & Computer Science,* vol. 7, no. 11, 2015.

Annex A: CPSoSaware Architecture as PlantUML code

```
@startuml
scale max 2000 width
left to right direction

title CPSoSAware - Components & Interfaces

/' Packages and components '/

package "CARLA Simulator" as carla {
    component VL [
        **ISI - T3.1**
        TC3.1.1 Visual Localization
    ]
    component DMSU [
        **ISI - T3.1**
        TC3.1.2 Deep Multimodal Scene Understanding
    ]
    component AIACC [
        **ISI - T3.1**
        TC3.1.4 AI Acceleration
    ]
    component MLAPI [
        **ISI - T3.1, T3.3**
        TC3.3.1 Multimodal Localization API
    ]
    component PPAPI [
        **ISI - T3.1, T3.3**
        TC3.3.2 PathPlanning API
    ]

    DMSU ..> MLAPI : provides input
    AIACC ..> DMSU : accelerates
    MLAPI ..> PPAPI : provides input
}

package "Semantic Components" as semantics {
    component SKG [
        **CTL - T4.5**
        TC4.5.1 Semantic Knowledge Graph
    ]
    component SKGS [
        **CTL - T4.5**
        TC4.5.2 Semantic Knowledge Graph Service
    ]

    () "SPARQL\nEndpoint" as sparql_endpoint
    SKG -down- sparql_endpoint
    SKGS ..> sparql_endpoint : queries
}
```

```
component SSA [
    **USI - T3.5**
    TC3.5.1 CPS Layer Security Sensors/Agents
]
component SRM [
    **Atos - T4.3**
    TC4.3.1 Security Runtime Monitoring
]
component V2X [
    **i2CAT T4.4**
    TC4.4.1 V2X simulator
]
component UBM [
    **UPAT - T3.1**
    TC3.1.3 User Behaviour Monitoring
]
component XRISA [
    **UPAT - T3.4**
    TC3.4.1 XR Tools for Increasing Situational Awareness
]

package "OpenCL platform" as opencl_platform {
    component PR [
        **TAU - T2.2, T3.2**
        TC2.2.2 pocl-remote
    ]
    component MLHAIP [
        **UoP/TAU - T2.3**
        TC2.3.1 ML Hardware Accelerator IP Cores
    ]
    component MTTO [
        **UoP - T2.3, T3.6, T4.6**
        TC2.3.3 Model transformation to OpenCL
    ]
    component XXKM [
        **UoP - T2.4, T3.6, T4.1**
        TC2.4.1 Xilinx XRT KPI monitoring
    ]
    component PACCEL [
        **TAU - T3.2**
        TC3.2.1 pocl-accel
    ]
    component TCE [
        **TAU - T3.6**
        TC3.6.1 TCE (openasip.org) soft cores
    ]
    component OCLWH [
        **TAU - T4.1**
        TC4.1.1 OpenCL Wrapper for Hardware IP Cores
    ]
    component PROFILING [
        **UoP - T4.1**
        TC4.1.2 Profiling
    ]
    component HLS [
        **UoP - T5.1**
        TC5.1.1 HLS-based SW to HW transformation
    ]

    TCE ..> PR
    TCE ..> OCLWH

    () "OpenCL API" as opencl_api
    PACCEL -up- opencl_api
    PR ..> opencl_api : uses
    OCLWH ..> opencl_api : uses
    MLHAIP ..> opencl_api : uses
    MTTO ..> opencl_api : uses
    XXKM ..> opencl_api : uses
    PROFILING ..> opencl_api : uses
    HLS ..> opencl_api : uses
}

component ICSIMT [
    **UoP - T2.2**
    TC2.2.1 Intra-Communication Sim Tool
]
component ICM [
    **UoP - T4.2**
    TC4.2.1 Intra-Communication Manager
]
database "Storage" {
    component IBMST [
        **IBM**
        Data Storage and Transformation Server
    ]
}
component MOT [
    **8BELLS - T2.5**
    TC2.5.1 Modeling Orchestration Tool
]
component AVSIM [
    **Robotec - T4.4**
    TC4.4.3 AV Simulation
]

/' Interfaces '/

queue "RabbitMQ" as skgs_bus {
}
DMSU ..> skgs_bus : publish data
DMSU <.. skgs_bus : receive reports
UBM ..> skgs_bus : publish data
SKGS <.. skgs_bus : consume data
SKGS ..> skgs_bus : publish reports

() "REST API" as path_planning_api
PPAPI - path_planning_api
SRM ..> path_planning_api : uses
V2X ..> path_planning_api : uses

() "REST API" as localization_api
MLAPI - localization_api
SRM ..> localization_api : uses
V2X ..> localization_api : uses

() "API (TCP 41000)" as srm_api
SRM - srm_api
SSA ..> srm_api : post sensor data

queue "RabbitMQ/Kafka" as srm_bus {
}
SRM ..> srm_bus : publish alarms

() "API" as xrisa_api
XRISA - xrisa_api
UBM ..> xrisa_api: Facial landmarks (2D coordinates), EAR, PERCLOS, yawings counter
UBM ..> xrisa_api: 3D coordinates of potholes and obstacles, occupancy factor
UBM ..> xrisa_api: Skeleton landmarks (2D coordinates) with confidence rates

() "REST API" as ibmst_api
IBMST -up- ibmst_api
ICSIMT <.. ibmst_api: receives inputs
ICSIMT ..> ibmst_api: stores outputs
ICM <.. ibmst_api: receives inputs
ICM ..> ibmst_api: stores outputs

() "Jenkins Pipeline" as mot_jenkins
MOT -down- mot_jenkins
ICSIMT <.up. mot_jenkins : uses
ICM <.up. mot_jenkins : uses
AVSIM <.up. mot_jenkins : uses
V2X <.up. mot_jenkins : uses
carla <.. mot_jenkins : uses

AIACC ..> UBM : accelerates
AIACC ..> PR : accelerates

@enduml
```