



## Horizon 2020 Program (2014-2020)

A computing toolkit for building efficient autonomous applications leveraging humanistic intelligence  
(TEACHING)

### D1.2: TEACHING CPSoS architecture and specifications<sup>†</sup>

Contractual Date of Delivery	31/08/2021
Actual Date of Delivery	27/09/2021
Deliverable Security Class	Public
Editor	Konstantinos Tserpes (HUA)
Contributors	<b>UNIPi:</b> Davide Bacciu, Daniele Mazzei <b>HUA:</b> Konstantinos Tserpes, Iraklis Varlamis, Ioannis Kontopoulos, Teta Stamati <b>CNR:</b> Alberto Gotta <b>TUG:</b> Jürgen Dobaj, Georg Macher <b>AVL:</b> Omar Veledar <b>I&amp;M:</b> Lorenzo Giraudi <b>M:</b> Calogero Calandra <b>TRT:</b> Sylvain Girbal <b>ITML:</b> Mina Marmpena <b>IFAG:</b> Jakob Valtl
Quality Assurance	Emanuele Carlini (CNR)

<sup>†</sup> The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871385.

## The TEACHING Consortium

University of Pisa (UNIPi)	Coordinator	Italy
Harokopio University of Athens (HUA)	Principal Contractor	Greece
Consiglio Nazionale delle Ricerche (CNR)	Principal Contractor	Italy
Graz University of Technology (TUG)	Principal Contractor	Austria
AVL List GmbH	Principal Contractor	Austria
Marelli Europe S.p.A.	Principal Contractor	Italy
Ideas & Motion	Principal Contractor	Italy
Thales Research & Technology	Principal Contractor	France
Information Technology for Market Leadership	Principal Contractor	Greece
Infineon Technologies AG	Principal Contractor	Germany

## Document Revisions & Quality Assurance

### Internal Reviewers

1. Emanuele Carlini (CNR)

### Revisions

Version	Date	By	Overview
1.0	27/09/2021	Editor	Final
0.4	25/09/2021	Reviewer	Comments on draft
0.3	23/09/2021	Editor	First draft.
0.2	07/09/2021	Contributors	Contributions
0.1	28/07/2021	Editor	ToC.

## Table of Contents

<b>LIST OF TABLES .....</b>	<b>5</b>
<b>LIST OF FIGURES .....</b>	<b>6</b>
<b>LIST OF ABBREVIATIONS.....</b>	<b>7</b>
<b>EXECUTIVE SUMMARY .....</b>	<b>8</b>
<b>1 INTRODUCTION .....</b>	<b>9</b>
1.1 RELATIONSHIP WITH OTHER DELIVERABLES .....	10
<b>2 HUMANISTIC INTELLIGENCE.....</b>	<b>12</b>
<b>3 TEACHING CPSOS APPLICATION MODEL.....</b>	<b>14</b>
<b>4 TEACHING PLATFORM &amp; NON-FUNCTIONAL REQUIREMENTS.....</b>	<b>16</b>
4.1 TEACHING PLATFORM INTRODUCTION .....	16
4.2 NON-FUNCTIONAL REQUIREMENTS .....	17
4.2.1 Safety.....	17
4.2.1.1 Quantified Requirements: Safety Assessment.....	18
4.2.2 Acceptance .....	20
4.2.2.1 Quantified Requirements.....	20
4.2.3 Real time performance .....	20
4.2.3.1 Quantified Requirements.....	21
4.2.4 Energy efficiency.....	21
4.2.4.1 Quantified Requirements.....	22
4.2.5 Cybersecurity .....	22
4.2.5.1 Quantified Requirements.....	23
4.2.6 Dependable AI.....	23
4.2.6.1 Quantified Requirements.....	24
<b>5 TEACHING USE CASES AND FUNCTIONAL REQUIREMENTS.....</b>	<b>26</b>
5.1 USE CASE #1: AUTOMOTIVE .....	26
5.2 USE CASE #2: AVIONICS .....	27
5.3 FUNCTIONAL REQUIREMENTS .....	28
<b>6 DESIGN-LEVEL DECISIONS.....</b>	<b>29</b>
6.1 ESTABLISHED TECHNOLOGIES.....	29
6.2 DESIGN DECISIONS.....	30
6.3 DESIGN DECISION PER FUNCTIONAL REQUIREMENT (FR) .....	32
6.3.1 FR1: Integration with the DMU.....	32
6.3.2 FR2: Sensing .....	32
6.3.3 FR3: Communication with external services .....	33
6.3.3.1 Intermittent communication scenarios.....	35
6.3.4 FR4: Application deployment.....	37
6.3.5 FR5/FR6: Model Training and Inference.....	38
<b>7 TEACHING PLATFORM ARCHITECTURE .....</b>	<b>40</b>
7.1 IMPLEMENTATION RECOMMENDATIONS.....	43
<b>8 CONCLUSIONS.....</b>	<b>47</b>
<b>9 REFERENCES .....</b>	<b>48</b>

## List of Tables

Table 1: Section map .....	10
Table 2: Deliverable grouping for verification of TEACHING Milestone 1 .....	11
Table 3: Safety Assessment.....	18
Table 4: Acceptance Assessment .....	20
Table 5: Real Time functional requirement.....	21
Table 6: Energy Efficiency Requirements .....	22
Table 7: Cybersecurity Requirements .....	23
Table 8: Dependability Requirements .....	25
Table 9: Summary of functional requirements .....	28
Table 10: Summary of TEACHING Platform Requirements .....	29
Table 11: Technology selection based on requirements.....	30

## List of Figures

Figure 1: Depiction of the IIRA Viewpoints from and mapping of focus of TEACHING Deliverables MS2 .....	11
Figure 2: Reciprocal relationship between human and computer in humanistic intelligence .....	12
Figure 3: A TEACHING Application closes the feedback loop between the human and the DMU ....	14
Figure 4: TEACHING Platform within the context of a system and its environment .....	16
Figure 5: Automotive CPSoS Application .....	26
Figure 6: Avionics CPSoS Application.....	27
Figure 7: High-level view of design decisions: Separation of underlying HW boards and communication with external services .....	31
Figure 8: Sensors API design .....	33
Figure 9: Example of the multiaccess communication and computing infrastructure for the automotive use case in Teaching.....	34
Figure 10: Overlapped scenario .....	36
Figure 11: Non-overlapped scenario .....	37
Figure 12: AIaaS SW Architecture Diagram, current design (source: D4.2: “Report on integrated mockup of the AIaaS system”).....	38
Figure 13: TEACHING platform conceptual architecture .....	41
Figure 14: Summary of TEACHING Platform Architecture .....	43

## List of Abbreviations

<b>EC</b>	<b>European Commission</b>
<b>WP</b>	Work Package
<b>e.MMC</b>	Embedded MultiMediaCard
<b>SDF</b>	Sensor Data Fusion
<b>PEC</b>	Physiological, emotive, and cognitive
<b>SOTIF</b>	Security of the intended functionality
<b>ML</b>	Machine Learning

## **Executive Summary**

The objective of this deliverable is to provide a definition and a design for an application-agnostic TEACHING Platform. Given the generic nature of the platform, we resort in providing a high-level component-level architecture and implementation recommendations that should be adapted by potential use cases. Technical-oriented work packages can use this design and instantiate it in order to deliver implementations that are tailored to the project use cases.

The proposed architecture design is derived a) from the project vision and most importantly the concept of Humanistic Intelligence; b) from the relevant non-functional requirements, and; c) from the functional requirements. Analyzing those allowed us to make some key-design decisions and base our architecture on top of a well-established baseline tools and technologies.

The remaining work needs to follow closely the implementations of the proposed architecture and use them to evaluate a set of KPI defined for each one of the requirements.



# 1 Introduction

This document provides a detailed account of the work conducted in the frame of “WP1: Requirements, Design and Integration of the Human-Aware CPSoS”. WP1 operates as an overarching WP to the technical WPs (WP2-5) providing directions and aggregating input from them with a final goal to formalize the concepts of the TEACHING CPSoS. The objectives of WP1 are to:

- Review the state-of-the-art, track future technology trends, and identify appropriate hardware platforms, sensors and software tools.
- Identify and track end user and technical requirements, querying use case partners and technical contributors of TEACHING.
- Specify the overall architecture for the TEACHING CPSoS by identifying components, their functionalities and interconnection, ensuring their coherency with the requirements and global architecture.
- Integrate the components/prototypes developed by the technical WPs and setup the integration environment in order to deliver the overall integrated TEACHING CPSoS and its demonstrators.

This document comes as a continuation of the report “D1.1: Report on TEACHING related technologies SoA and derived CPSoS requirements” which was delivered on M12 (31/12/2020). The first two objectives have already been covered by D1.1 and this report (D1.2) is focusing on the effort to provide the design and specifications for the TEACHING platform, using the results reported in D1.1. That is, the focus of the work reported here is targeting the 3<sup>rd</sup> objective and as such, it attempts to provide a definition of the architecture and implementation recommendations for an application-agnostic TEACHING Platform.

A starting point to this analysis was the TEACHING project goal as defined in the Description of Action (ref appendix to the Grant Agreement):

*The goal of the TEACHING project is to design a computing platform and the associated software toolkit supporting the development and deployment of autonomous, adaptive and dependable CPSoS applications, allowing them to exploit a sustainable human feedback to drive, optimize and personalize the provisioning of their services.*

Based on that we distinguish two main artefacts:

- TEACHING Platform
  - A computing platform
  - A development and deployment software toolkit
- TEACHING Applications
  - Autonomous, adaptive and dependable CPSoS applications

A main focus of this document is to further define these concepts in the context of the project vision and especially that of humanistic intelligence.

The document reports the efforts to provide those definitions in the form of design patterns and recommendations. Those can serve as a guidance to the Research and Development (R&D)

work of the project in its effort to deliver its promised outcomes for the particular application domains.

For the remaining of the document, we will refer to the computing platform and the software toolkit collectively as the **TEACHING Platform** which is meant to support the development and deployment of the **TEACHING applications**.

In order to fulfil the T1.2: “Architecture design” objectives, the participating partners built on the analysis of the TEACHING CPSoS (**system**) **requirements** already defined in the work of T1.1: “Requirement identification”.

To convey the results of this work, this report starts with a theoretical definition of the humanistic intelligence (Section 2) and then uses it to define the TEACHING CPSoS Applications (Section 3). The latter is a central step in the identification of the requirements and the definition of the platform generic properties (referred to as “modules”). Those modules are used for the sketching of a conceptual TEACHING Platform and they relevant non-functional requirements (Section 4). A similar analysis for the elicitation and update of the relevant functional requirements is done on the basis of the project use cases (Section 5). Leveraging on the functional and non-function requirements analysis we present the design decisions (Section 6) which finally lead to the presentation of the TEACHING Platform architecture and implementation recommendations provided to WP2-5 (Section 7).

To simplify the reading of this document we provide the following table, explaining what are the objectives of each of the section.

**Table 1: Section map**

Section	Title	Objective/Outcome
Section 2	Humanistic Intelligence	Introduction to the Humanistic Intelligence overarching concept
Section 3	TEACHING CPSoS Application model	Umbrella definition of the TEACHING CPSoS and Platform core modules
Section 4	TEACHING Platform and non-functional requirements	Analysis of non-functional requirements for the Platform
Section 5	TEACHING Use cases and functional requirements	Definition of Platform functional requirements on the basis of the project use cases
Section 6	Design-level decisions	Presentation and justification of key-design decisions
Section 7	TEACHING Platform architecture	Presentation of Platform architecture and implementation recommendations

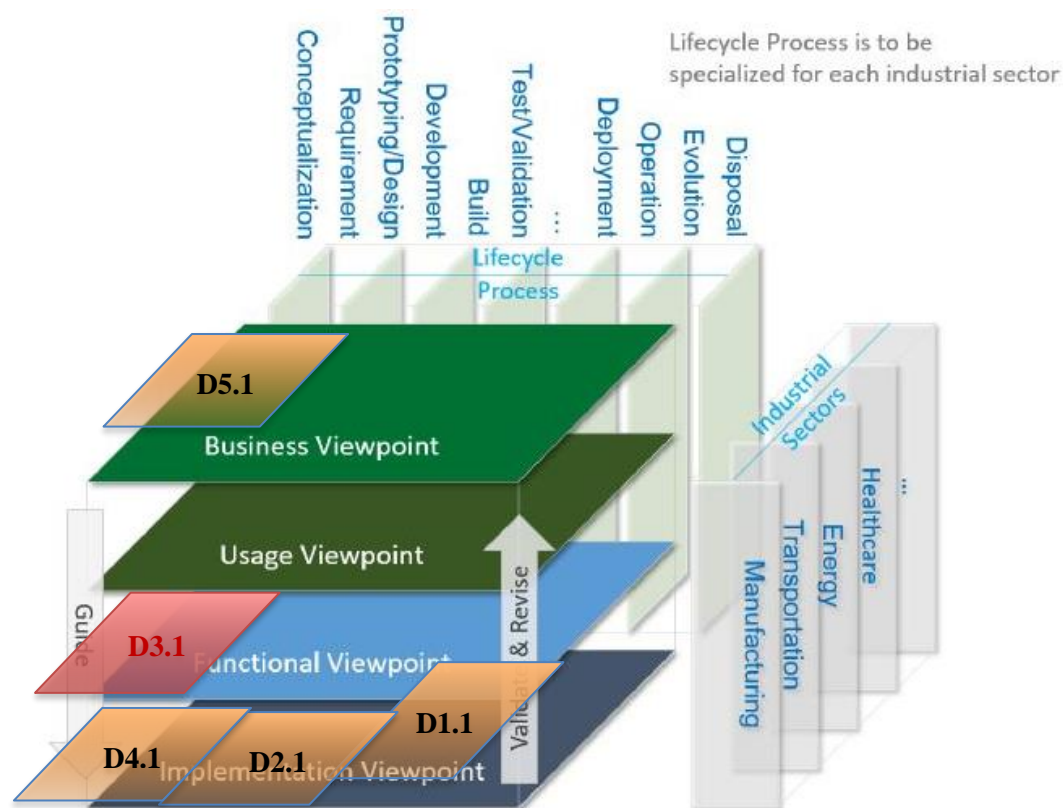
## 1.1 Relationship with other deliverables

There is a group of related deliverables, i.e., D1.2, D2.2, D3.2, D4.2 and D5.2 (Table 2), all of which serve as a mean of milestone MS2 verification. That is the second project milestone, entitled “*First integrated setup with mock-up of the TEACHING platform*”.

**Table 2: Deliverable grouping for verification of TEACHING Milestone 1**

D1.2	TEACHING CPSoS architecture and specifications
D2.2	Refined requirement specifications and preliminary release of the computing and communication platform
D3.2	Interim Report on Engineering Methods and Architecture Patterns of Dependable CPSoS
D4.2	Report on integrated mockup of the AIaaS system
D5.2	Preliminary use case deployment, implementation and integration report with related dataset release

The mapping of the viewpoints of the technical WPs, as well as the integration intentions of the TEACHING technology bricks in domain use-cases is depicted in Figure 1.



**Figure 1: Depiction of the IIRA Viewpoints from <sup>1</sup> and mapping of focus of TEACHING Deliverables MS2**

<sup>1</sup> <https://iiot-world.com/industrial-iiot/connected-industry/iic-industrial-iiot-reference-architecture/>

## 2 Humanistic Intelligence

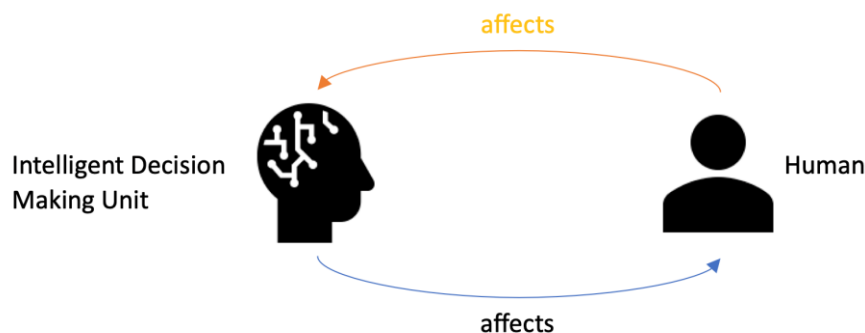
Humanistic Intelligence (HI) is intelligence that arises because of a human being in the feedback loop of a computational process, where the human and computer are inextricably intertwined. In the context of TEACHING the design of a CPSoS architecture centered on the concept of humanistic intelligence becomes mandatory.

The TEACHING application of HI focuses on the development of a CPSoS able to match and adapt biological brain capabilities and needs. More generally TEACHING CPSoS focuses on the creation of an AI-powered system that results from a feedback loop between a computational process and a human being, where the human and computer are inextricably intertwined.

In the field of human-computer interaction (HCI) it has been common to think of the human and computer as separate entities. HCI emphasizes this separateness by treating the human and computer as different entities that interact. However, HI theory thinks of the interacting human and the computer with its associated input and output facilities not as separate entities, but regards the computer as a second brain and its sensory modalities as additional senses, in which synthetic synesthesia merges with the human's senses. When a wearable computer or an enclosing machine (airplane, car and any other system that includes and transport human beings) functions in a successful embodiment of HI, the computer uses the human's mind and body as one of its peripherals, just as the human uses the computer as a peripheral. This reciprocal relationship is at the heart of HI (Figure 2).

Essentially, the classic human-in-the-loop approach typical of HCI is here reframed and addressed as a design problem. The questions we decided to answer in the design of the TEACHING CPSoS architecture are: “how do we incorporate useful, meaningful human interaction into an AI-powered system?”, “how we leverage human intelligence for improving the AI?”.

This kind of human-centered design approach is at the center of research in fields like Interactive Machine Learning, in which intelligent systems are designed to augment or enhance the human, serving as a tool to be wielded through human interaction. We can see this type of research expressed in the works of Alison Parrish, poetical engineer, as well as at the Stanford HAI launch event in March 2019, which highlighted collaborative social systems; computers that learn to help and many other humanistic intelligence application examples (more info [here](#)).



**Figure 2: Reciprocal relationship between human and computer in humanistic intelligence**

Humanistic Intelligence is applied in Teaching by designing a CPSoS architecture that intrinsically enable HI. This is achieved by a continuous supervision of all the design activities that aims at guaranteeing that human is always considered as a pillar and central element of the entire system. This supervising activity has been conducted in Task 1.3: “Human-centered design”.

### 3 TEACHING CPSOS Application model

With HI theory in mind, the goal shifts to the definition of the TEACHING Platform and of its design and specifications. Therefore, we need to first understand what the platform application requirements are, i.e., we need to define the TEACHING Applications. To do so, we start with the underlying concepts, i.e., the definition of the TEACHING CPSoS as it was formulated in D1.1.

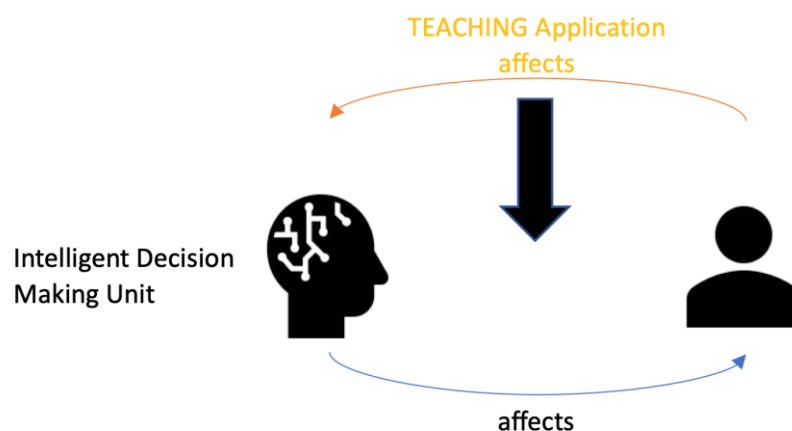
Assume a cyber-physical system (CPS) whose behaviour is regulated by a feedback control loop. This system can be a vehicle or an airplane and its behaviour ranges from simple navigation to emergency manoeuvres. This system is stable/dependable in the sense that it achieves its objectives which are dictated by concrete rules/policies, under a wide range of conditions. TEACHING introduces a new objective to the control loop, through the integration of a new, potentially non-dependable system. This system brings along a new array of sensors for monitoring its state, different than then ones that the original CPS is using. The new integrated system of systems (SoS) extends the original CPS and it remains a CPS itself. We will refer to it as the **TEACHING CPSoS**. As a CPS, it needs to maintain the properties of CPSs.

Following the principles of the humanistic intelligence concept the TEACHING CPSoS is instantiated by introducing a human in the loop as follows.

We assume that there is a machine that is intelligent enough to make decisions autonomously. We will refer to that machine as a Decision-Making Unit (DMU). The DMU decisions affect the state of a system, e.g., a vehicle or airplane. Those decisions are made to attain some preset objectives about the system state. That is, the DMU is trying to maintain the prescribed system state(s).

We now introduce a human as part of this system. The system's state now is extended to include the human state. In such a way, the human is affected by the DMU but also affects it, closing the feedback loop.

A **TEACHING Application** is the mechanism that enables the integration of the human state to the system state (Figure 3).



**Figure 3: A TEACHING Application closes the feedback loop between the human and the DMU**

Based on the above, a quick conclusion is that a platform that would support a TEACHING Application should be able to support at least the monitoring and quantification of the human state and its integration to the DMU. We consider those three properties as the basic functional requirements for the TEACHING CPSoS Platform dubbed “TEACHING Platform”. To ensure that the platform maintains its general-purpose nature, we add one more property: programmability. In detail,

- **Monitoring:** The platform must be able to monitor the human state using appropriate sensors and possibly considering external factors, such as the environment to which the system is operating
- **Quantification:** The platform must be able to model the human state and quantify it in a way that is meaningful to the DMU
- **Integration:** The platform must be able to incorporate the quantified human state in the system state so as for the DMU to use it.
- **Programmable:** The platform must provide an interface that will enable the dynamic creation of TEACHING applications.

## 4 TEACHING Platform & non-functional requirements

### 4.1 TEACHING Platform introduction

The TEACHING Platform is the environment that enables the TEACHING Applications to close the feedback loop. The business actors of the TEACHING Platform were identified in D1.1 (Section 3.1), namely, the owners and the users of the platform and their agents, i.e., platform and application providers and platform administrators and application developers.

Apart from those roles, there are two central human actors in a TEACHING Application that the TEACHING Platform should accommodate their needs: the TEACHING Application developer and the human whose feedback needs to be incorporated in the loop. Figure 4 presents the high-level building blocks of the TEACHING Platform and its interaction with the two human actors as well as with the DMU and by extension, the system.

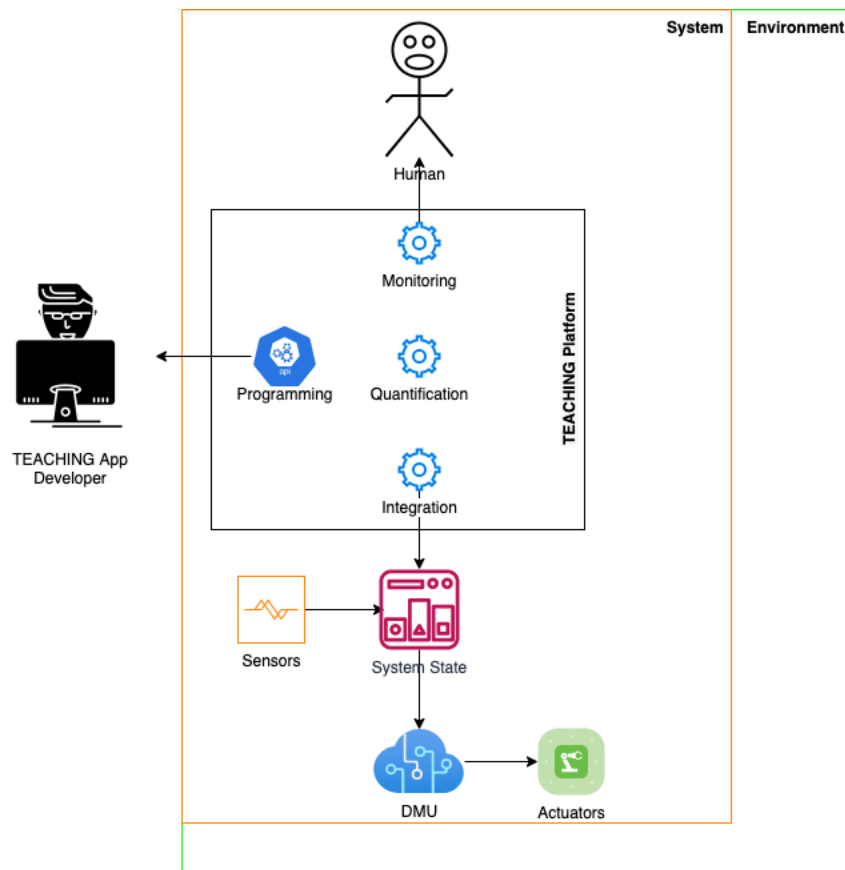


Figure 4: TEACHING Platform within the context of a system and its environment

The figure depicts the 4 basic modules of the TEACHING Platform and the fact that three of them (**Programming, Monitoring and Integration**) are interacting with the actors and system. The Monitoring module interacts with the Human in the system, collecting data that will allow the **Quantification** module to model the human state. The Integration module then, integrates the human state to the system state that feeds the DMU to control the actuators and manage the



system. The sensors that create the primary system state may or may not take into consideration the environment.

This flow could be functional as it appears, however there are certain constraints that dictate certain non-functional requirements to be taken into consideration should the TEACHING Platform be considered a seed for a viable product.

Among the primary constraints is the fact that the TEACHING Platform is an addition to an existing system rather than a built-in feature. This implies that the TEACHING Platform must not affect the expected functionality of the original system. This functionality is crucial to preserve the systems' properties that cannot be compromised. One such crucial property is safety. We need to find engineering approaches that guarantee that the integration of human state in the system state does not lead to safety metrics degradation.

Furthermore, the resources upon which the DMU is operating may be limited. This fact may compromise the CPS's real-time performance. The TEACHING platform may infuse a delay in the CPS's operations that cannot be afforded and therefore the integration of the states must happen in a way that will not affect normal performance.

On a similar note, the TEACHING Platform may impact the CPS by limiting its lifetime due to energy constraints. The CPS may operate on limited energy resources making the execution of the TEACHING platform or its operations on the same energy source impossible.

Finally, we need to address the modules interaction with the human and ensure that it happens in a user-friendly way. This has to be considered also from a commercial point of view, i.e., if a new feature requires an obtrusive method to operate, then the customers will never embrace it.

Those non-functional requirements are further elaborated in what follows.

## **4.2 Non-functional requirements**

In this section we narrow down the discussion about the TEACHING Application non-functional requirements (NFRs) by explaining how they affect the TEACHING Platform. To quantify the latter, we define metrics to be followed for the assessment of the architecture design and implementation.

### **4.2.1 Safety**

Safety is itself an application of design rules and requirements useful to accomplish several functionalities at vehicle level by ensuring always absence of unreasonable risk due to hazards resulting from performance limitation or malfunctioning behaviour of safety-related E/E systems. However, in this respect, Safety by design can be viewed as an essential tool (as an NFR) to achieve a dependable product through the definition of the technical safety concept which contains all foreseen functional requirements. In order to facilitate the ability to verify the system architectural design, the technical capability of the intended hardware and software elements regarding the achievement of functional safety, and the ability to execute tests during

system integration, safety requirements have to be clear and precise, with a layered definition based on the different design abstraction level that goes from system to HW and SW.

The alignment of the Security Of The Intended Function (SOTIF) and Functional Safety is fundamental to implement possible modifications to the system design at a sufficiently early stage. The Identification and Evaluation of Triggering conditions as initiator for a hazardous behaviour, considers system limitations and evaluates possible functional modification to reach an acceptable SOTIF risk according to the definition of Technical Safety Concept of ISO 26262<sup>2</sup> processes. In addition to a safe design and development process, assessment should carry on iteratively from verification to validation and comprise safety analyses and experiments.

#### 4.2.1.1 Quantified Requirements: Safety Assessment

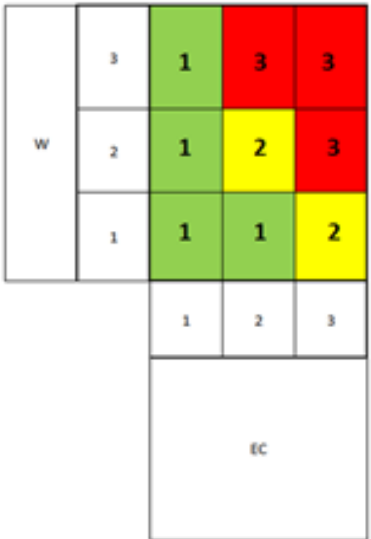
As mentioned in the standard ISO 26262 and ISO 21448, principles of safety for automated driving provide a foundation for deriving a baseline for the overall safety requirements and activities necessary for the different automated driving functions. Based on state-of-art, the interpretation of the principles of safety with respect to AI -based functions should be fully explored in accordance with the upcoming standard ISO-PAS 8800; the field of AI, and in particularly its use for critical applications, is developing very rapidly. ISO-PAS 8800 should describe system and function-specific evidence provided to support a system level safety assurance case commensurate to the principles of ISO 26262 and ISO 21448 as well as other emerging standards such as ISO TS 5083.

Hence, each functional safety requirement should be defined taking into account a set of main principles as safety baseline. Particularly, the safety principle for **Safety Assessment** traces to all safety measures that a CPSoS should have. This derives from the expectation that product development would be responsible for delivering a necessary level of evidence for the verification and validation of the safety measures, which may then be reviewed by an assessing group.

**Table 3: Safety Assessment**

ID	Title	Description	means of verification (KPI)
NFR1	Functional Safety Assessment	Evaluate the status of the activities/work products related to the different parts of the ISO 26262 safety-lifecycle and their availability, with respect to the maturity of the project foreseen by the Safety Plan and Project Plan in order to achieve the safety compliance	RISK matrix = W *EC

<sup>2</sup> <https://www.iso.org/obp/ui/fr/#iso:std:iso:26262:-3:ed-2:v1:en>

			 <p>W: Weight of the findings or issues according to the work products:</p> <ul style="list-style-type: none"> <li>- Formal finding scores 1</li> <li>- Process finding scores 2</li> <li>- Product finding scores 3</li> </ul> <p>EC: Real completeness wrt expected contents completeness:</p> <ul style="list-style-type: none"> <li>- Completeness scores 1</li> <li>- Incompleteness scores 2</li> <li>- Missing contents scores 3</li> </ul> <p>Assessment/Risk Scoring rules:</p> <ul style="list-style-type: none"> <li>- Full compliance scores 1 (product development accepted)</li> <li>- Partial compliance scores 2 (product development with conditional acceptance)</li> <li>- Unsatisfactory compliance scores 3 (product development rejected)</li> </ul> <p>Note: for the final product release is necessary to reach the product development acceptance (full compliance)</p>
NFR2	SOTIF Assessment	Evaluate the status of the activities/work products related to the different parts of the ISO 21448 safety-lifecycle and their	RISK matrix = W *EC (see KPI ID.001)

		availability, with respect to the maturity of the project foreseen by the plan for SOTIF activities and Project Plan in order to achieve the safety compliance	
--	--	--	--

#### 4.2.2 Acceptance

The automotive industry's shift towards driving automation is a gamble that can only pay off if there is an appropriate level of user acceptance so that the end-users can reap real benefits from the offered technical solutions. As digitalisation is in its full swing, the noticeable effect is not only that of the product development, but also the need for the provision of services, many of which are data-based or data-reliant. TEACHING fits in this equation by developing the appropriate set of algorithms and tools to support the shift towards driving automation. While studies suggest a strong connection between acceptance and trust in technical solutions [Dimi20] and [Mol18], TEACHING also has an additional advantage when it comes to the development and integration of new technology solutions. The advantage stems from the project's focus on human-centric control of the safety-critical systems. As the control is intended to objectively respond to the user subjective perception of the environment, the expectation is that through continual usage of the offered solutions (integrated into the appropriate industrial environment) the user acceptance is to increase. Such increase has already been demonstrated through the improvement of trust in the automated system [Cle21].

##### 4.2.2.1 Quantified Requirements

Table 4: Acceptance Assessment

ID	Title	Description	means of verification (KPI)
NFR3	Acceptance levels	Quantify the subjective ratings of overall satisfaction with the driving simulator experience and determine the increase of user acceptance level.	Quantification of the driving simulation satisfaction level through subjective user questionnaires and correlation of the obtained answers to the (objective) psychophysiological measurements.

#### 4.2.3 Real time performance

Hard real-time applications are characterized by stringent real-time constraints at process or task level, expressed as expected schedule properties and termination deadlines. As a consequence, correctness of an operation is not only defined by functional correctness, but also by time-window during which those operations or tasks have to be executed.

The recent shift to multi-core processor, and now the shift to heterogeneous architectures with AI accelerators is introducing new sources of time variations identified as **timing interference**.

Consequently, the industry is facing a trade-off between performance and predictability [Kir08], [Mez11].

Several studies [Bin14], [Now12] have shown that the maximum observed execution time slowdown on a multicore / heterogeneous environment could be much higher than the expected benefits of using such architectures. Also, the common practice of determining Worst Case Execution Time (WCET) relying on static program analysis tools [Wil08], detailed hardware model, as well as measurement techniques through execution or simulation [Hec05] is not currently able to deal with multi-core and heterogeneous architectures.

Timing and resource consumption guarantees are therefore especially required for AI accelerators and their associated software, with hard real time guarantees especially for the inference part of the algorithm that requires to be executed online on the embedded board.

#### 4.2.3.1 Quantified Requirements

If the hard real time requirements are very hard to guarantee as required by the certification standards. Detecting deviation from the requirements is much easier to detect as they are reflected by **deadline misses** at task or process levels. Additional deadlines need to be expressed at AI level, especially for the inference part so that deadline misses could also be detected at that level that is not necessarily captured by the RTOS scheduler.

**Table 5: Real Time functional requirement**

ID	Title	Description	means of verification (KPI)
NFR4	Real-Time	Guarantee hard real-time scheduling policy and deadlines of periodic and aperiodic tasks.	Number of deadline misses not increased due to monitoring and AI features

#### 4.2.4 Energy efficiency

Energy efficiency is a key point in CPSoS. Especially setups where the systems are deployed on the edge. Usually not only energy resources in the form of batteries or other power supplies are limited. Also the hardware parts are planned to be as small as possible and as large as necessary to meet the requirements. This holds for all hardware parts like computing, storing and sensing as well as actuator hardware. Specialized hardware like GPUs, FPGAs, ASICs or other accelerators are, if integrated in the CPSoS, an ideal way to maximize energy efficiency if the tasks carried out by the CPSoS can make use of them. Once the hardware is defined a great deal of energy efficiency can be gained by software optimization. During the investigation on IFAG's demonstrator with a special focus on the software optimization with the purpose of systems energy efficiency localized the following topics as most relevant for CPSoS energy efficiency with regard to software optimization:

- Reduction of schedule frequency, thus the update rate with which repetitive tasks are carried out to operate at the lowest frequency possible, while still meeting system operation requirements.

- Develop software with a version control system that includes testing on unnecessary components to reduce the operating system to its needed functionality. This way tasks and functionality that is outdated and not needed in the latest version will not be implemented on the edge and reduce storage and computation needs.
- Intelligent tests that look for functionality that is processed various times throughout the system at different points can lead to an optimization, thus combining, executing the operation only once and storing the result globally, thus reducing the processing needs. In this way, unnecessary and repetitive calculation of code can be avoided.

#### 4.2.4.1 Quantified Requirements

Quantification of energy efficiency is usually most interesting when compared to other systems performing the same task or earlier versions. The main driver for energy efficiency is usually the limitations of the hardware or power resources or external specifications like timing or other constraints.

**Table 6: Energy Efficiency Requirements**

ID	Title	Description	means of verification (KPI)
NFR5	Energy Efficiency	Determine the energy efficiency in terms of measurable parameters and compare available parameters to benchmark alternatives or earlier versions.	-overall power consumption compared / related with -memory / CPU usage -data throughput -timing parameters

#### 4.2.5 Cybersecurity

Artificial intelligence (AI) can be used to foster security in CPSoS by providing proactive security mechanisms to detect both well-known attack patterns, but also predict unknown ones. In TEACHING, AI-powered components can provide solutions for monitoring and applying dependability constraints, i.e., an anomaly detection module can be used to monitor network traffic, detect abnormal patterns, and estimate their critical status to inform a human or decide an action on each own. In that sense, the AI component monitors other system components and their communications, makes inferences about the normality of their behaviour, and takes actions to attain the dependability of the system – either with a human feedback or based on its own policies or models. Deep learning algorithms used for anomaly detection are not restricted by specific rules or malware signatures, and thus, they can potentially generalize better since they are essentially built to learn expressive representations of complex data and model nonlinear relationships within the data domain, instead of applying specific rules. In other words, an AI-based solution at inference, will look for patterns of malicious events, instead of hard-coded instances. In domains where the task and the data dimensionality are of high complexity, deep learning algorithms can offer a scalable solution.

The current anomaly detection algorithm implemented as a module for TEACHING AIaS API is a Long Short-Term Memory Autoencoder (LSTM-AE) [Mal17]. The model follows from the

conceptual paradigm of learning feature representations of normality [Pang20] to detect anomalies. Essentially, this paradigm couples feature learning and anomaly scoring in a unified model. This is achieved by learning representations which are forced to capture underlying regularities from normal data using a generic objective function. The anomaly scoring is then calibrated based on the reconstruction error of the normal representation, and values exceeding it are indicators of anomalies. The generic nature of the algorithm makes it suitable for other applications besides cyber threat detection within the TEACHING framework. For example, anomaly detection can be applied to human biometrics data, or resources consumption measurements, runtimes, and other unlabelled time-series data.

#### 4.2.5.1 Quantified Requirements

For the evaluation and verification of the anomaly detection model the Key Performance indicator selected is the *recall* rate metric on benchmark datasets. Low anomaly detection recall rate poses a challenge to a plethora of anomaly detection methods that suffer from a high false positive rate of detected anomalies. Thus, a high recall rate is one of the most important and yet difficult challenges, especially because it might be achieved at the expense of failing to detect enough truly anomalous events.

**Table 7: Cybersecurity Requirements**

ID	Title	Description	means of verification (KPI)
NFR6	Security: Integrity	Data streams are monitored with an anomaly detection module to detect rare or abnormal events and unexpected patterns. The model used is a Long Short-Term Memory Autoencoder evaluated on labelled benchmark datasets.	$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$ TP: True positives FN: False negatives

#### 4.2.6 Dependable AI

The use of data-driven AI techniques as building blocks of ICT systems needs to be carefully assessed from a dependable system engineering perspective, especially when targeting applications that interface and interact with humans, either on the cyber or on the physical world. As this is the reference scenario for applications that will be supported by the integrated TEACHING platform, its development process needs to carefully account for requirements associated to ensuring dependability of the TEACHING AI components. These requirements need to target the specificity of the learning models integrated into the AI-as-a-service toolkit which are elements of the Recurrent Neural Networks (RNNs) family. The choice of this class of learning models has been driven by the fact that they are designed to model sequential information, which is the prevalent data type in a CPSoS. As an additional level of detail, TEACHING designs and develops RNNs from the Reservoir Computing paradigms, where the recurrent component implementing the dynamic neural memory is randomized and untrained.



The two “Dependable AI” dimensions that are explored here, and that complement the cybersecurity perspective from the previous section, are Safety and Privacy. Safety pertains to the containment of the risks associated to harmful consequences of system operation, on itself, on the environment and the humans involved. Safety considerations in neural networks are complicated by their inherent black box nature, which makes inspection of their inner workings not easily accomplishable, especially by humans. Recurrent NNs are, in this sense, even more complex and difficult to inspect than feedforward models, since their behaviour is defined by the history of the inputs presented to the network, through their dynamic memory, rather than from a single current stimulus. As a consequence, an RNN can fail in unpredictable ways. Therefore, the enforcing of safety properties on systems incorporating RNNs poses two requirements: (i) the availability of a methodology that can identify the normal operational regimes of the learning model and (ii) the availability of mechanism controlling when the RNN operates outside of the normal regimes and that applies the necessary harm prevention strategies. In order to address the requirement associated to point (i) above, TEACHING will provide a method to measure the robustness of the RNNs integrated in its AIaaS toolkit with respect to inputs perturbations, such as those generated by systematic errors in sensors data, by environmental conditions, or adversarial perturbations, using state-of-the-art methods such as POPQORN [Ko19]. The information from the robustness measure allows to perform a plausibility check on the operation of the RNNs in step (ii), resulting in corrective actions by a fallback system in case of detection of an erratic/untrusted behaviour. Privacy pertains instead with the preservation of secrecy of sensitive user data. A key requirement for the TEACHING AI components is, in particular, that of avoiding leaks of private information used for the purposes of learning models personalization. Again, the implementation of this requirement is targeted to the specificity of the learning models used in the AIaaS. To this end, TEACHING provides privacy-aware training mechanisms based on the differentially private SGD framework [Aba16]. Other cybersecurity features aside of privacy will also be considered in the context of work package 3 of the TEACHING project.

#### **4.2.6.1 Quantified Requirements**

In contrast to traditional applications that are explicitly programmed or defined through human-understandable rules, machine learning techniques instead enable computers to learn tasks by using data. The established safety development processes and practices (described by, e.g., ISO 26262 and ISO/PAS 21448) in the automotive industry have been only successfully applied in traditional model-based system development. However, none of the available safety standards within the automotive and any other industry have defined processes that explicitly consider the specifics of machine learning approaches like the requirements on dataset collections, the definition of performance evaluation metrics, the handling of uncertainty, etc [BMW19]. In addition, the increased sharing of data and the machine learning approaches themselves introduce new cybersecurity risks to the overall system.

The ultimate goal in the engineering and development of dependable systems is to maximize the evidence of a positive risk balance. In the given context, the TEACHING project aims to provide evidence that the risks introduced by the increasing demand on connectivity and data sharing and the risks to violate system safety through using machine learning approaches are



adequately addressed. To that purpose, structured approaches for the risk-driven development of machine learning approaches into safety-related applications shall be proposed.

**Table 8: Dependability Requirements**

<b>ID</b>	<b>Title</b>	<b>Description</b>	<b>means of verification (KPI)</b>
NFR7	Positive Risk Balance	Provide evidence that the TEACHING platform exhibits a positive risk balance.	<ul style="list-style-type: none"><li>- Risk analysis and risk assessment of the TEACHING platform design.</li><li>- Proposal of risk mitigation measures or statement of risk acceptance.</li></ul>

## 5 TEACHING Use Cases and Functional Requirements

In this section we introduce the two project use cases filtered through the concepts of the TEACHING Application and TEACHING CPSoS as they have been defined above. We are focusing on identifying the 4 modules from Figure 4 and through those to identify the high-level system functional requirements. To achieve that, we start by presenting the narrative pertaining to the main scenarios of the project use cases.

### 5.1 Use case #1: Automotive

The developer wants to implement an application that performs online learning based on human feedback to feed a vehicle's Advanced Driver-Assistance System (ADAS) and have it adapt to the comfort levels of the passengers (Figure 5). The latter is monitored through a set of sensors and their input is “translated” to metrics that quantify the passengers' stress levels by a model. This information is aggregated with situational awareness data (environment and vehicle status) in another model that delivers an adjustment towards the ADAS that is fit to the passenger's preferences.

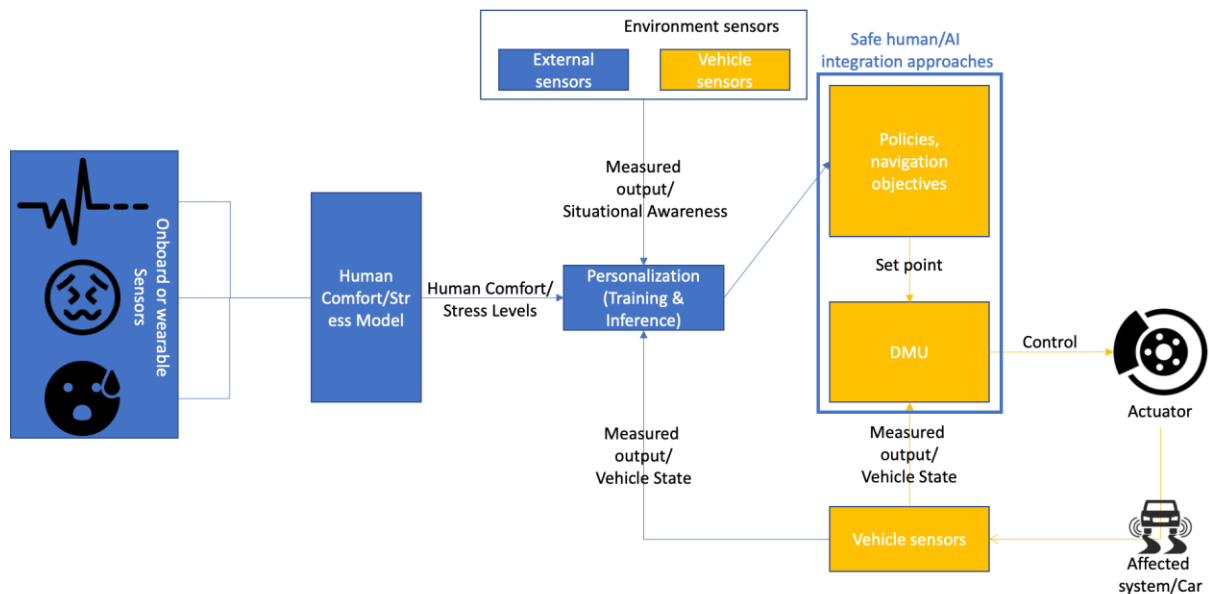


Figure 5: Automotive CPSoS Application

The application depicts the functional components implementing each of the four core modules:

- **Monitoring:** Onboard and/or wearable sensors monitor the human state as the vehicle's (system) state and the environment (road, weather, etc.) change. The latter two are monitored by vehicle sensors (e.g., temperature) and/or external services (e.g., traffic or weather services) and in-vehicle sensors (e.g., accelerometer).
- **Quantification:** The Human Comfort/Stress model quantifies the sensor signals into a measurable unit of human comfort or stress. This function is largely based on AI/ML tools.
- **Integration:** The output of the quantification step is fed into the control loop of the system, the DMU, through another model that uniquely maps the passengers

comfort/stress levels to a preferred style of driving. This model is the Personalization component and it operates on a reinforcement learning principle. The Personalization component essentially selects preset driving styles that the DMU offers.

- **Programming:** Although not apparent, it is expected that some tools should allow the deployment of models for the quantification and integration steps, as well as the data workflows.

## 5.2 Use case #2: Avionics

The developer wants to develop an application that implements and executes an anomaly detection DL algorithm so as to detect high stress levels on the hardware on top of which the software of the Flight Management System (FMS) of an aircraft is executed (Figure 6). The application must also provide mitigation proposals to the pilot. The application is comprised of a set of probes that monitor the hardware that runs the FMS. The monitoring data are aggregated and summarized in a stress levels metric. This metric, along with the onboard sensors are then processed with the intention to identify anomalies (anomaly detection). Potential anomalous events are then reported to the pilot, along with mitigation plans. The actuation of these mitigation plans can alter the state of the plane, as well as the computational effort on the hardware, in turn affecting the operation of the DL algorithms. In this case, the human enters the loop in an offline way, providing indications about appropriate actions to be taken by the DMU.

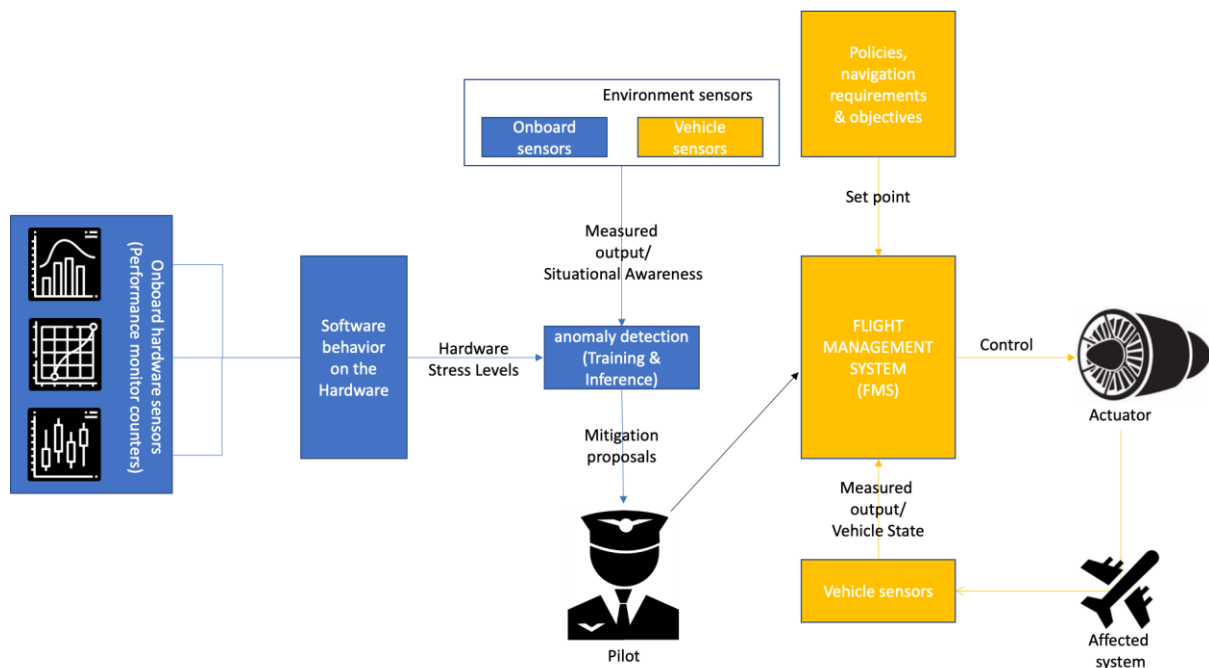


Figure 6: Avionics CPSoS Application

In a similar fashion as in the previous use case, the functional components are mapped into the four core modules of the TEACHING Platform.

- **Monitoring:** Onboard sensors monitor the human reactions (humanistic state) as the airplane's FMS (system) state and the environment (weather conditions, etc) change.

The latter two are monitored by hardware sensors (e.g., cache misses) and onboard sensors (e.g., gyroscope).

- **Quantification:** The software behavior on the hardware is monitored and quantified through the onboard hardware sensors as indicators of the hardware wear. At the same time, the human reaction to various events and wear levels is interpreted into a correctness scale. This function is largely based on AI/ML tools.
- **Integration:** The output of the quantification step is fed into the control loop of the system, the FMS (DMU), through another model that uniquely maps the onboard hardware situation to an appropriate pilot action. This model is based on Anomaly Detection techniques.
- **Programming:** Similarly, as in UC #1, the programming element is not apparent. However, it is again expected that some tools should allow the deployment of models for the quantification and integration steps, as well as the data workflows.

### 5.3 Functional requirements

The discussion above indicates a link between the four conceptual modules of the TEACHING Platform and a number of functional requirements. Summarizing those requirements for the two use cases we resort to a list presented in Table 9. The Table also refers to the method to assess the successful implementation of each of the requirements in a similar fashion as with the non-functional requirements (ref Section 4.2).

**Table 9: Summary of functional requirements**

Module	ID	Title	Description	means of verification (KPI)
Integration	FR1	Integration	integrate with the DMU and interact with it	Testing
Monitoring	FR2	Sensing	communicate with the wearable sensors	Testing
	FR3	External_services	communicate to external components and services through the web	Testing
Programming	FR4	App_Deployment	support a runtime for the deployment of software components	Testing
Quantification	FR5	Model_train	support ML/AI algorithms' training	Testing
	FR6	Model_run	support ML/AI algorithms' inference	Testing

## 6 Design-level decisions

Summarizing the above, the **TEACHING Platform is a computing platform and a software toolkit** that must implement a defined set of functional and non-functional requirements. Design decisions need to be taken based on these. For the reader's convenience the requirements are summarized in Table 10.

**Table 10: Summary of TEACHING Platform Requirements**

ID	Title	Description
NFR1	Functional Safety Assessment	Evaluate the status of the activities/work products related to the different parts of the ISO 26262 safety-lifecycle and their availability, with respect to the maturity of the project foreseen by the Safety Plan and Project Plan in order to achieve the safety compliance
NFR2	SOTIF Assessment	Evaluate the status of the activities/work products related to the different parts of the ISO 21448 safety-lifecycle and their availability, with respect to the maturity of the project foreseen by the plan for SOTIF activities and Project Plan in order to achieve the safety compliance
NFR3	Acceptance levels	Quantify the subjective ratings of overall satisfaction with the driving simulator experience and determine the increase of user acceptance level.
NFR4	Real-Time	Guarantee hard real-time scheduling policy and deadlines of periodic and aperiodic tasks.
NFR5	Energy Efficiency	Determine the energy efficiency in terms of measurable parameters and compare available parameters to benchmark alternatives or earlier versions.
NFR6	Security: Integrity	Data streams are monitored with an anomaly detection module to detect rare or abnormal events and unexpected patterns. The model used is a Long Short-Term Memory Autoencoder evaluated on labelled benchmark datasets.
NFR7	Positive Risk Balance	Provide evidence that the TEACHING platform exhibits a positive risk balance.
FR1	Integration	integrate with the DMU and interact with it
FR2	Sensing	communicate with the wearable sensors
FR3	External_services	communicate to external components and services through the web
FR4	App_Deployment	support a runtime for the deployment of software components
FR5	Model_train	support ML/AI algorithms' training
FR6	Model_run	support ML/AI algorithms' inference

In what follows we further elaborate on these design decisions on a per functional requirement basis, starting with the baseline technology selection that we have made before providing the generic TEACHING Platform architecture.

### 6.1 Established technologies

To reduce the risk of failing to meet those requirements, we build upon a slightly revised list of established technologies, already identified in project report D1.1: "Report on TEACHING

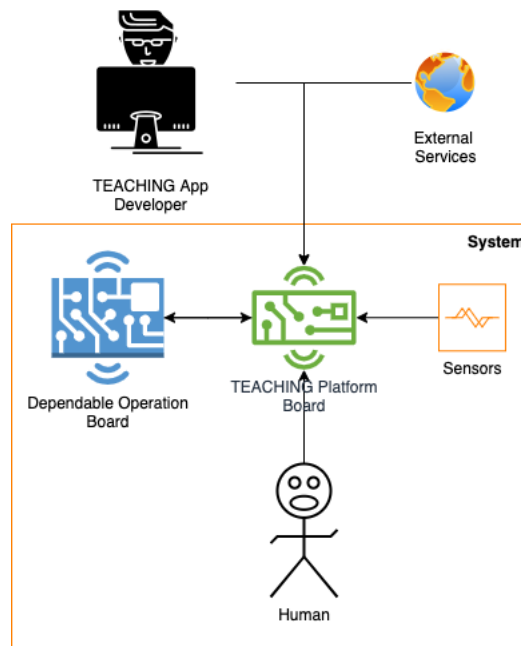
related technologies SoA and derived CPSoS requirements”. This selection is reflected on Table 11 with the addition of a column linking to the relevant requirements, i.e, to the requirements that are meant to be satisfied by the TEACHING Platform.

**Table 11: Technology selection based on requirements**

<b>TEACHING Platform aspect</b>	<b>Technology selected</b>	<b>Relevant requirement</b>
Hardware Platform	I&M SDF for the automotive and iMX8 for the avionics use case	FR1-6, NFR4,5,7
Sensors for hardware monitoring	METriCS	NFR3,4
Sensors for human monitoring	Shimmer array, plus emotive and respeaker sensors, onboard cameras	FR2
Resource orchestration and execution environment	Docker swarm or K3S	FR4,6
ML/DL libraries/toolkits	TensorFlow/Lite	FR5,6
IoT libraries/toolkits	OS-IoT, MQTT broker, Kafka broker	FR2,3
Security libraries/toolkits	OpenSSL (low level), NIST lightweight encryption	NFR6
DB libraries/toolkits	SQLite	FR4
Communication	Gigabit Ethernet	FR1

## 6.2 Design decisions

One of the first design decisions was to separate the hardware on top of which the TEACHING Platform and the applications run with the hardware on top of which the dependable system runs (Figure 7). This is reflected in the first row of Table 11 and further elaborated in the project report D1.1. The key idea is to run the 3 basic modules of the TEACHING Platform (ref: Section 3): Programming, Monitoring and Quantification on the TEACHING board and work on a standard communication method for the Integration.



**Figure 7: High-level view of design decisions: Separation of underlying HW boards and communication with external services**

This approach shifts the burden for meeting the Safety requirements to the dependable system board which is expected to be compliant to NFR1&2 by default (from the original vendor). Furthermore, it separates the design concerns regarding real-time performance, security and dependability (NFR4,6,7) on the TEACHING Platform.

Of course, devices at the edge of the CPSoS are typically embedded devices that have intrinsic limitations that have to be carefully considered. The consortium selected the SDF from I&M as the TEACHING Platform board as the one that navigates a good tradeoff between all NFRs and FRs. In what follows we further explain the rationale behind this decision.

In the case of the chosen platform, the storage is limited by the size of the eMMC of 32GB. This means that only a limited amount of data can be downloaded to the board and at the same time. At the same time, if data is being gathered to be sent to the cloud, attention should be paid to the length of the record.

Moreover, the i.MX8 Quad Max features 2 Cortex A72 and 4 Cortex A53 in a big.LITTLE configuration. The manufacturer reports some performances based on the Coremark benchmark, which evaluates the performances of the CPU. The reported value is 13933.78 iteration/sec while performing the benchmark on 4 cores out of 6. To have the overall performance, the Coremark has been executed on the SDF using 6 cores: the result is 26038.58 iteration/sec. Another limitation is given by the main memory: on the platform, there are 6GB of LPDDR4 RAM running at 1.6GHz. It has to be managed carefully to avoid swaps from/to the storage that would decrease the performance and introduce unpredictable delays.

The i.MX8 Quad Max features two GC7000XSVX GPUs from Verisilicon. Unfortunately, they do not support the well-known CUDA language used by Nvidia GPUs, but they still support OpenCL and other graphic libraries like OpenGL ES, OpenVX, and others. Moreover, NXP provides the abstraction layer needed to run TensorFlow Lite on the GPU.

Because of the resource-constrained nature of the edge platform, some design choices are limited. For instance, more lightweight solutions have to be chosen if available, and for HW-dependent software, it is necessary to check the present support. Both reasons lead to the choice of Tensorflow Lite as a machine learning framework, being both lightweight and natively supported. Moreover, every solution has to be tested on the target board to understand the impact on resource usage.

### **6.3 Design decision per functional requirement (FR)**

#### **6.3.1 FR1: Integration with the DMU**

In terms of Integration between the TEACHING Platform board and the dependable system board, the decision is to use a standard Gigabit Ethernet protocol which is natively supported by the selected boards. Another reason for this decision was to tackle interoperability, i.e., the fact that more mainstream technologies are preferred over specialized.

This latter point is placed quite centrally in the TEACHING design decision making. In order to allow the TEACHING platform to operate on multiple application domains and for multiple scenarios, the platform board must be based on mainstream technologies so as to maintain some generic properties.

Apart from the means for the integration between the two boards, there is also the crucial point of the logical integration. What API does the dependable system board offer that the TEACHING Platform could use so as to logically integrate with its operation?

This is an open-ended question, in the sense that there is no restriction in the implementation of the DMU and the possible interfaces (if they exist) commonly differ from one application domain to another. As such, the design decision is to allow the TEACHING Application developers to implement this part on their own. The recommendation that TEACHING makes based on a preliminary dependability analysis reported in D3.2: “Interim Report on Engineering Methods and Architecture Patterns of Dependable CPSoS”, is to pursue the implementation of mechanisms that would allow the selection of preset operations in the dependable system.

#### **6.3.2 FR2: Sensing**

Already in project report D1.1, we had identified six kinds of sensor information which can be useful for the TEACHING human monitoring purposes and for feeding the AI components responsible for human PEC state estimation:

- Inertial data
- Cardiac and respiratory data
- Myographic data
- Electrodermal activity
- Brain activity
- Sound from the passengers.

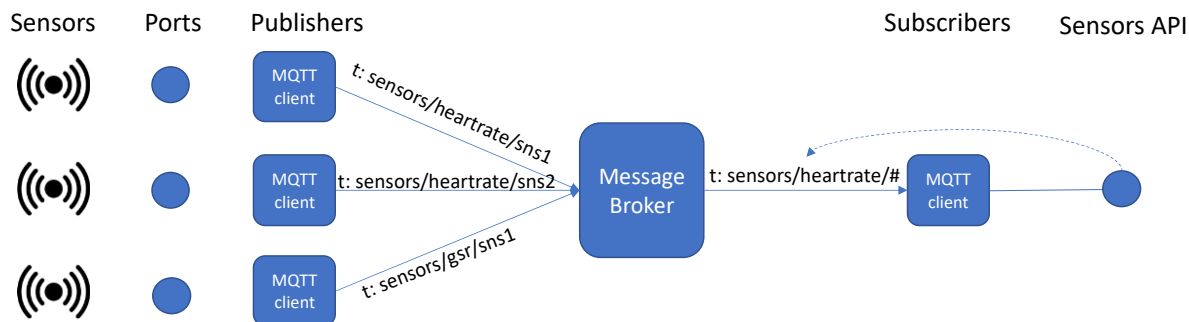


Following up the definition of the physical and physiological parameters to be monitored, we have identified a list of sensing devices enabling their collection. The selection of the specific devices has been driven by the following considerations:

- Maturity of the technology as assessed in terms of available software and technical support, as well as in terms of market readiness.
- Compatibility with existing communication technologies, operating systems, and software.
- Ease of configuration and personalization.
- Ease of integration of different devices with same/similar communication and synchronization protocols.
- Accessibility of raw sensor data with none/minimal pre-processing to allow full exploitation of sensed information from the data-driven AI models responsible for PEC estimation.
- Preferably devices that have already been validated for use in biomedical-oriented research applications.

In light of NFR3 (Acceptance), the usability of this array of wearable sensors is questionable, however they are the most appropriate to for modelling PEC. More suitable sensors in terms of acceptance are considered to be glasses for eye gazing tracking and on board time-of-flight (TOF), thermal and RGB cameras. The project will attempt to create models for the second type of sensors (cameras) employing transfer learning techniques from the first type (wearables).

A standard interface for the sensors to feed in the TEACHING Applications is required. As such, we designed an API for the Sensors based on a pub/sub pattern and the MQTT protocol for message brokerage (Figure 8).



**Figure 8: Sensors API design**

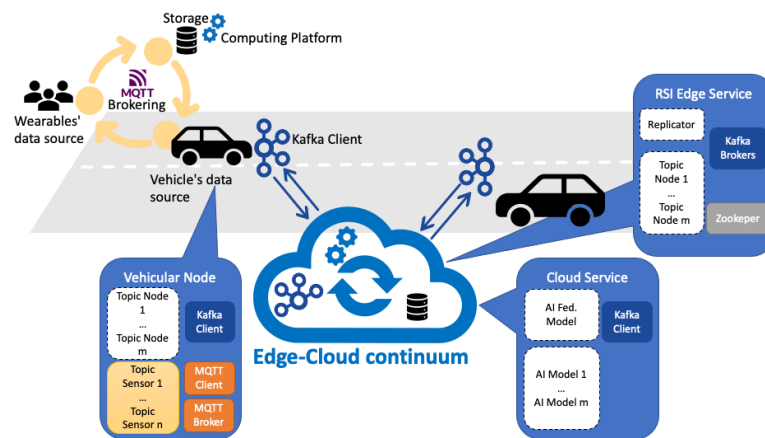
This design is further explained in D4.2: “Report on integrated mockup of the AIaaS system”. but it needs to be revised as it is not suitable for multimedia streaming as is. Support for camera content streaming can be achieved by simply using the message broker to relay information about the data stream that opens at the level of the OS.

### 6.3.3 FR3: Communication with external services

Vehicle-to-everything (V2X) communications represent an important niche in both networking and vehicular technologies fields. Sattiraju et al. in [Sat] compare link level performance of Intelligent Transport systems (ITS) -G5 standard, based on IEEE 801.11p, and the more recent Cellular-V2X (C-V2X), which is based on 3GPP standards. As stated in [Sat], IEEE 802.11p standard is a well matured technology that has been researched for over 20 years.

Notwithstanding this, the standard never had a deep impact in the automotive mass market, probably due to the major investment in communication infrastructure to support ITSs. A comparison with legacy LTE networks was already presented in 2014 in [Mol], while the Release.12 LTE Device-to-Device (D2D) link level put the comparison of the two standards on a fairer footing. C-V2X, introduced in Release.14, sheds light on V2X communications, with the first specification in 3GPP standards and Sattiraju et al. in [Sat] achieve that C-V2X outperforms IEEE 802.11p for almost all the considered fading channels with a gain ranging from 0-5 dB, and, above all, at high vehicle speeds. C-V2X extends LTE's D2D communication modes by defining two operation modes for vehicular systems: (i) Mode 3 in the coverage and (ii) Mode 4 out of the coverage of an LTE eNB, which is the radio base station of an LTE network [Mar].

By now, we generally refer to a Road-Side Unit (RSU) with the generic radio communication device belonging to the roadside infrastructure, which, in general, can coincide with an eNB in Mode 3 but not limited to. The equivalent notion in UC#2 is the gate or the hangar where the airplane parks.



**Figure 9: Example of the multiaccess communication and computing infrastructure for the automotive use case in Teaching**

The communication scenario is exemplified through the scheme shown in Figure 9 and is fully described and detailed in D2.2. For convenience, this is reported here to detail the communication issues linked to such a scenario. It highlights three main entities: i) Vehicular Node, ii) Road-Side Infrastructure (RSI) Edge service and iii) Cloud service. The mobile node is equipped with an MQTT broker<sup>3</sup> to publish, locally, the data produced by the vehicle's sensors. Data is published into a set of topics, such as: *private\_data*, *shared\_data* and *AI\_model*. Data destined to in-vehicle processing can be published onto the *private\_data* topic; data that can be shared with other vehicles or edge/cloud services can be published onto the *shared\_data* topic. Finally, the model topic is devoted to data concerning the *AI\_model*

<sup>3</sup> <https://mqtt.org>

shared among the entities of the infrastructure. Data topics are accessed by both local and remote subscribers.

The computing architecture designed in TEACHING assumes that edge servers are co-located in the RSI with the RSUs. The entities of the RSI resulting from such a combination provide to Vehicular Nodes both network access and computing capabilities. A topic replicator is installed in the RSI, which allows replicating broker's managed data (i.e., topics) over a set of federated brokers (the Kafka brokers<sup>4</sup> in Figure 9), allowing to increase the resilience of the RSI toward the possible faults of the deployed brokers.

Vehicular Nodes, moving on a vast geographical area, where such a communication infrastructure is deployed, will be always able to interact with the redundant brokers through the RSUs.

The Cloud Service is the most remote computing entity of the proposed infrastructure, from the perspective of the vehicular node, i.e., the data producer. This entity is devoted to achieving the federation of the AI models trained on the Edge. Generally speaking, the Cloud is not necessarily to be identified with public cloud services (e.g., AWS, Google Cloud, etc.), but can be associated with a private Cloud (e.g., a datacenter provided by automotive vendors).

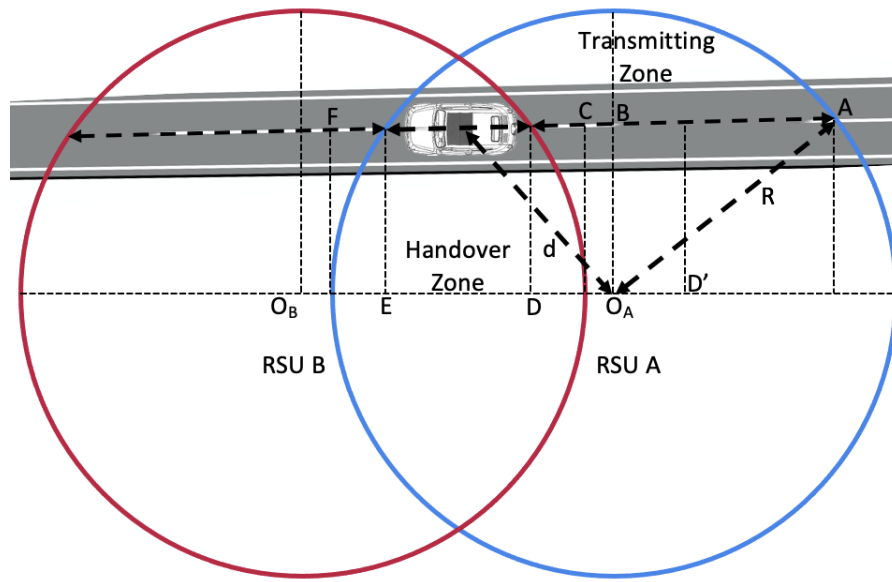
The algorithm that runs on Cloud performs the federation of the AI models trained on the Edge.

#### **6.3.3.1 Intermittent communication scenarios**

The use case presented in Figure 9 can be characterized into two communication scenarios in Figure 10 and Figure 11, which focus on the issue of intermittent connectivity. Figure 10 applies when the adjacent cells, i.e. the coverage area of an RSU, of two different RSUs overlap. Conversely, Figure 11 does apply when the aforementioned cells do not overlap.

---

<sup>4</sup> <https://kafka.apache.org>

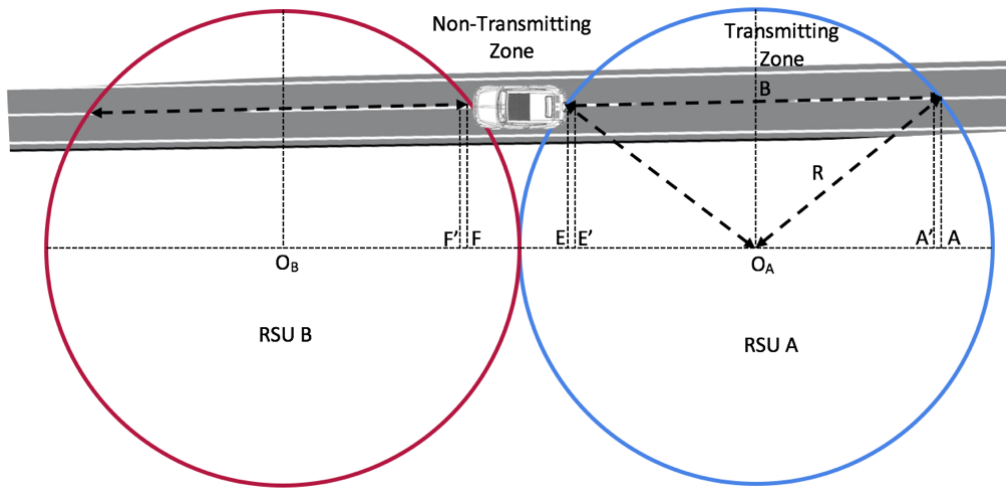


**Figure 10: Overlapped scenario**

Let's assume that vehicles move along the trajectory with constant velocity  $v$ , and that the linear density of vehicles per meter of the trajectory is  $\delta$ . In this mobility scenario, we also assume that the coverage range of the cell is  $R$  and the height difference of the antennas between the vehicle and the cell is  $\Delta h$ . The cells can overlap each other and the minimum distance between antenna and end-user is  $FC$ . The time required for the handover is  $T_{HO}$ .

For the scenario in Figure 10 the transmitting time interval  $T_{ON}$  is given by the length  $DA$  divided by the average vehicle speed  $v$ . Note that, at the time  $T_{ON}$ , we need to subtract the  $T_{HO}$  (handover time between the previous cell and the current one) and, in the case of the first encountered cell, also the time  $T_{CON}$ , which is required by a broker to establish a client-broker connection at application layer, after that the radio link is fully recovered (i.e. after  $T_{HO}$  seconds). When the mobile user is within the handover zone the interval time without transmitting data ( $T_{OFF}$ ) is given by:

$$T_{OFF} = \text{Min} \{T_{HO}, ED/v\}.$$



**Figure 11: Non-overlapped scenario**

Regarding the scenario in Figure 11,  $T_{ON}$  is given by the length  $E'A'$  divided by the speed  $v$ . Note that, we need to subtract the time  $T_{CON}$  at the time  $T_{ON}$ , again. In this scenario the disconnected time is given by,

$$T_{OFF} = F'E'/v.$$

$AA'$ ,  $EE'$ , and  $FF'$  are the space interval required to cover the radio distance  $R$ . During these small intervals, depending on the communication direction (uplink or downlink) the vehicle could exit the coverage area (not receiving data coming from the RSU) or could have to wait to start the connection with the cell.

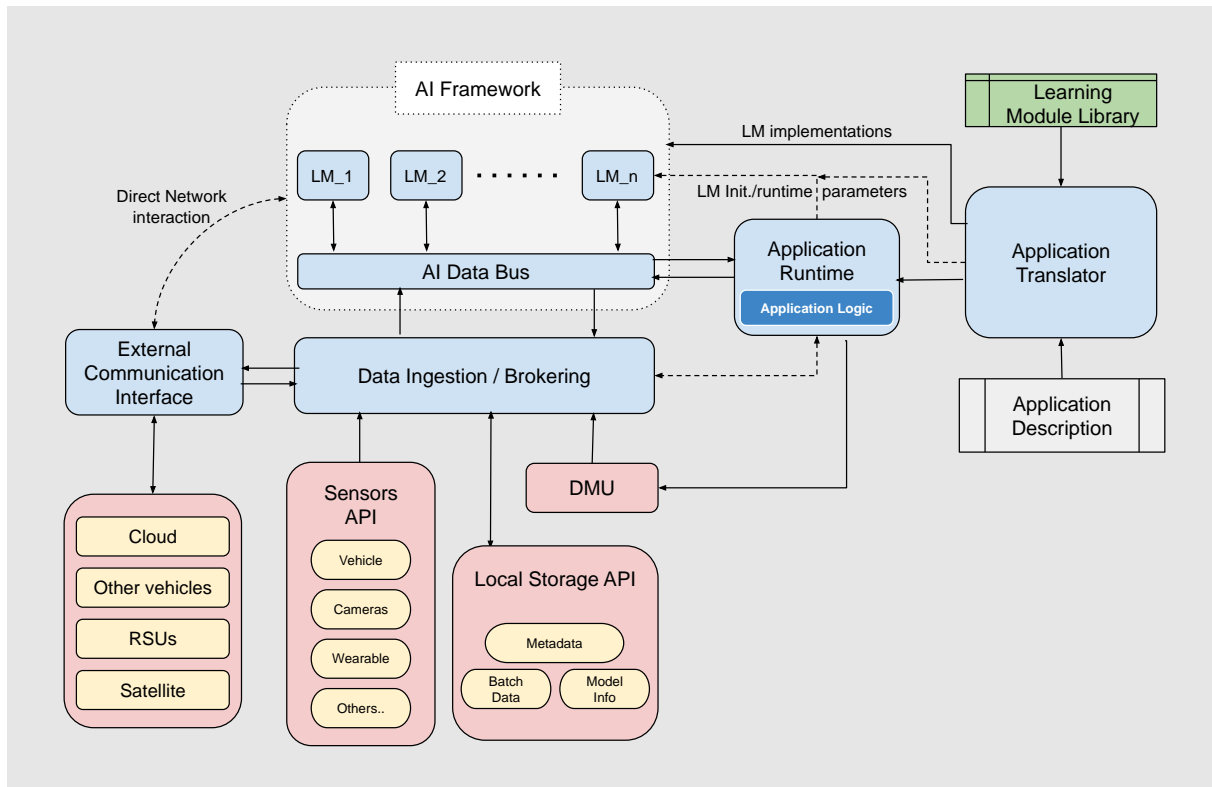
#### 6.3.4 FR4: Application deployment

The key design decision for implementing this requirement is the use of containers that can host application images. To further accommodate load balancing and fault tolerance requirements, we resorted to orchestrators. Some initial experimentation with K3S in various platforms (Jetson Nano, Raspberry Pi) yielded mixed results in terms of its suitability. On occasions, the orchestrator consumed more than 80% of the platforms CPU without any application image being loaded in the container. We resorted to the use of simple docker containers rather than orchestrators which resulted in better resource utilization, stripped from the benefits that an orchestrator may bring.

Further experiments were conducted in order to ensure that the containers were able to communicate with the sensors and that the applications (e.g., application models, ref subsection 6.3.5) can run exploiting the underlying OS capabilities and the infrastructure (e.g., GPUs). An application model was also developed allowing the developer to use an expressive syntax in a YAML format so as to program an AI-enabled application to run, creating data workflows and scheduling the training and inference phases of a variety of algorithms.

### 6.3.5 FR5/FR6: Model Training and Inference

This functional requirement is tightly coupled to the Quantification module. The human state needs to be modelled using the PEC state sensors. For that purpose, we need tools that will simplify the development for the application developer and that will efficiently use the underlying infrastructure (e.g. GPUs), given also its limitations in the form of NFRs. This has taken shape in the form of a service that is offered to the application developer for the creation and training of AI models, dubbed “AIaaS” (Figure 12).



**Figure 12: AIaaS SW Architecture Diagram, current design (source: D4.2: “Report on integrated mockup of the AIaaS system”)**

The AIaaS component is explained and presented in detail in D4.2: “Report on integrated mockup of the AIaaS system”. The same report, among others, provides the following rationale for the design choices for this component:

The TEACHING approach to AIaaS on Edge and Cloud devices relies on designing reusable, portable AI application as a combination of composable, generic app-building blocks called Learning Modules (LM) and data sources. The rationale of the approach is that:

- The LM building blocks can be separately ported to and optimized for different device HW/SW architectures, increasing their efficiency with respect to common metrics (performance, power consumption), allowing careful debugging and verification, as well as allowing to exploit specific features of the execution platform within the LM.
- AI applications are more easily developed, reducing their overall complexity, increasing their reliability, and shortening the time-to-market.

- AI applications effortlessly become as much portable as the LM supporting SW architecture is. That is, deploying apps on a plethora of Cloud and Edge devices is allowed by making the focused effort of adapting the AIaaS support to those devices, without need of changing the apps and allowing different HW/SW devices to interoperate in a distributed software platform.

The aim of developing a dedicated support architecture for AIaaS in TEACHING thus requires choosing a trade-off between LM expressiveness and tailoring the LM to the HW. This is necessary in order to strike a manageable balance between achieving reusability of AI Apps across AIaaS implementations and easing the porting of the whole AIaaS architecture to new devices (this shall remain a mostly straightforward and manageable task except possibly for HW-specific optimizations). Two main abstract goals were held as reference “*lighthouses*” in the process of architecture design:

- *Allow adoption in TEACHING (being fit for the use cases):* The architecture must be portable and lightweight to suit the automotive and avionic use cases, allowing to build generic application with the suite of LM provided, while at the same time allowing to exploit specific hardware resources thanks to interchangeable implementations of the same LMs.
- *Allow reuse in different contexts:* The AIaaS supporting architecture shall be useful as a tool for porting AIaaS applications in different execution contexts, including the Cloud and various types of Edge devices (e.g., mobile units as well as fixed edge devices). Developing a full AI stack and development kit would be out of scope and would not get any adoption, thus the architecture needs to be designed exploiting existing, technologically relevant and/or industrial-standard AI frameworks at its core, namely:
  - TensorFlow
  - TensorFlow Lite
  - WindFlow / FastFlow

## 7 TEACHING Platform architecture

One conceptual view of the TEACHING Platform was presented in the project report D1.1: “Report on TEACHING related technologies SoA and derived CPSoS requirements” (Figure 14). This conceptual architecture is following the rationale of layered architectures, where each layer offers services to the one above. Instantiations of the conceptual architecture may include implementations that merge layers, similarly as ISO/OSI and TCP/IP.

The starting point for designing the architecture of the TEACHING Platform is the TEACHING goal which states “a computing platform and the associated software toolkit supporting the development and deployment of autonomous, adaptive and dependable CPSoS applications”. As such, at the top layer we place the CPSoS applications that are meant to be supported by the computing platform and the software toolkit, i.e. the TEACHING Platform.

Based on our definition of CPSoS applications, i.e. the applications that meet a certain number of NFRs, we provide a layer whose components are meeting those NFRs. This layer is meant to provide the specification of the software toolkit.

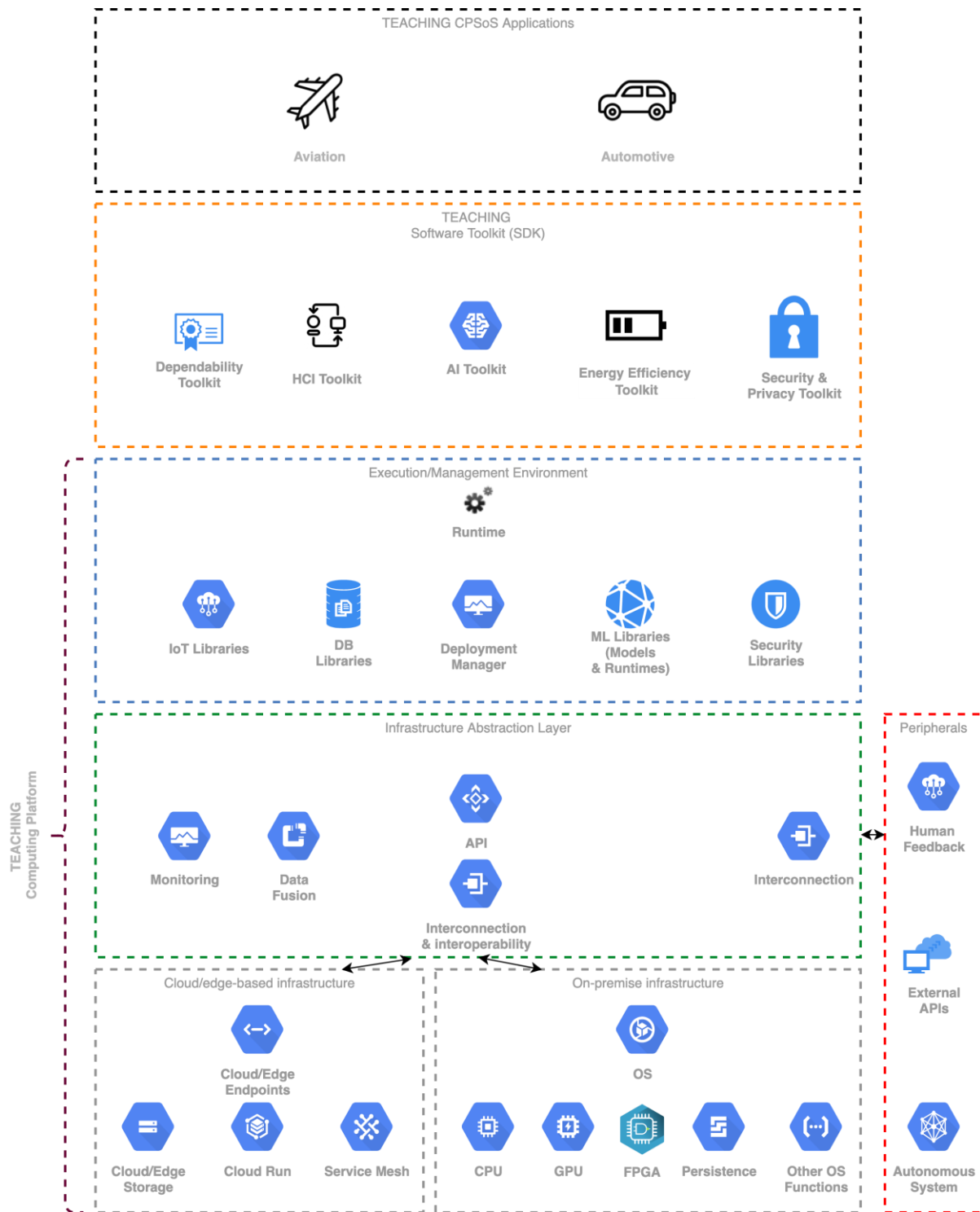
The underlying layers are forming the TEACHING computing platform. They start with the layer that is meant to provide all the supporting software tools that will allow the development of the CPSoS applications and meet the functional requirements.

The layer below is meant to specify the way that the computing platform will deal with interoperability issues, homogenizing the underlying computing and network infrastructures.

The final layer is dealing with the specification of the infrastructure.

In what follows, we provide a more detailed view of the TEACHING Platform.





**Figure 13: TEACHING platform conceptual architecture**

The **TEACHING platform** is comprised of 5 layers, each of which provides services to the one above. At the bottom of the stack, we have the infrastructure layer.

**Infrastructure Layer:** The infrastructure layer is comprised of various heterogeneous infrastructures, exposed through an embedded system OS and the cloud/edge resources. TEACHING assumes that access to the resources of those infrastructures is a priori possible.

On that premise, the first task of TEACHING is to homogenize those resources, something that is the main functionality of the Infrastructure Abstraction Layer.

**Infrastructure Abstraction Layer (IAL):** The IAL provides a single, abstraction layer for execution of applications (code or components). Essentially it homogenizes the underlying infrastructures providing a single API to deploy, execute and monitor resources and application components. This layer also caters for implementing I/Os, with the underlying persistence layers as well as with the supported peripherals, i.e., the target autonomous system (CPS), external APIs (e.g., web services), but most importantly with the mechanisms that provide the human feedback.

**Execution/Management Environment (EME):** The EME exposes a single API that facilitated the execution and lifecycle management of the application components. It provides the runtime for that purpose, along with integrated libraries, implemented at a low-abstraction language, providing services and optimizations at the top layers. Such libraries include ML runtimes such as those of Tensorflow and PyTorch, or ML optimizations in Python, C++, Java, etc. It also includes libraries for managing IoT solutions (e.g., OS-IoT) implementing IoT protocols such as OneM2M. Other libraries include the DB and security libraries ensuring that this kind of functionality is provided to the layers above.

**TEACHING Software Toolkit (SDK):** The TEACHING SDK provides the framework to implement CPSoS applications. It provides APIs to implement applications that can run on the TEACHING platform making the best use of the CPSoS services. The TEACHING SDK supports 6 toolkits:

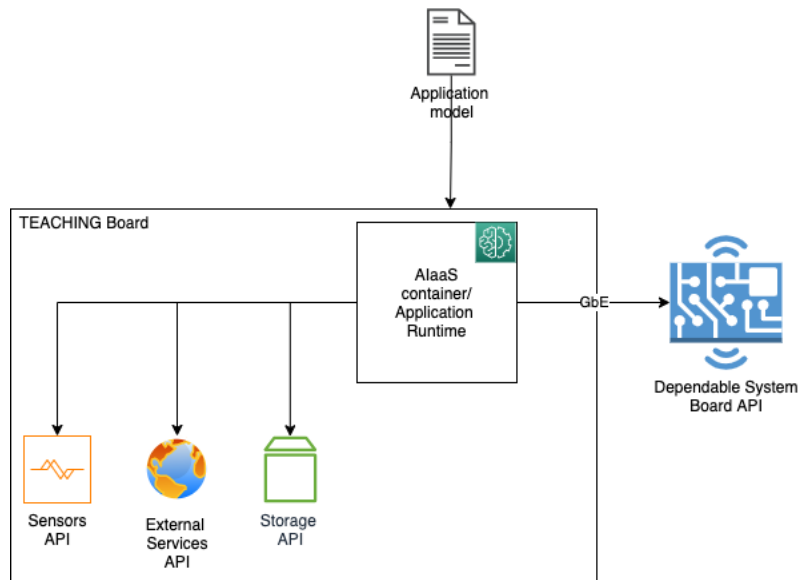
- The **AI toolkit** is the software library that allows the developer to invoke learning modules, set up training or inference procedures, etc. The AI toolkit has the appropriate wirings with the underlying layers to deploy and run the ML components at the appropriate resources (e.g., GPUs) and facilitates the I/Os and dataset management.
- The **HCI toolkit** allows the software developer to invoke the services that are relevant to the human feedback, e.g., filters, buffers and other suchlike tools for retrieving and managing the human feedback. Furthermore, this toolkit includes design patterns and guidelines for humancentered design.
- The **Security and Privacy toolkit** provides readily available security APIs as well as privacy guidelines. In terms of security, the developers may define a part of their code or a standalone component that has to run on a secure enclave, or that the communication between components has to use OpenSSL calls. In terms of privacy, the developers may identify datasets as containing sensitive data, thus implicitly imposing constraints in their further use. Furthermore, the privacy toolkit may also include functional tools like anonymizers.
- The **Dependability toolkit** provides software that audits the code or application components against the TEACHING dependability guidelines/procedures. It also provides engineering patterns implementations that the developers can invoke, for ensuring the dependable execution of software. For instance, in cases where the developers invoke online training approaches through the AI toolkit, the dependability toolkit may allow the code to run in multiple instances implementing a consensus model.
- The **Energy Efficiency toolkit** is linking the code or components that the user would like to run with energy efficiency services provided by the underlying layers. E.g., in

order to run an application, the toolkit may employ energy efficient approaches such as dynamic voltage and frequency scaling (DVFS), power mode management (PMM) or using unconventional cores such as DSP or GPUs of FPGAs. This can be done automatically or invoked by the user (e.g., “annotating” a part of the code or a component).

**TEACHING CPSoS Applications:** The TEACHING applications may be comprised of loosely coupled, standalone, independent components (e.g., docker images) that the TEACHING SDK builds or software that the TEACHING SDK compiles and executes.

## 7.1 Implementation recommendations

Following the discussion about the design decisions which followed the conceptual architecture, we resort to a simplified version of the TEACHING Platform architecture (Figure 14). This is focusing on a TEACHING Platform that is comprised of the TEACHING Board (I&M SDF) and a software kit implementing the functional and non-functional requirements. Instantiations of the components of this architecture with implementation details are provided in D2.2 and D4.2 while some design decisions are further elaborated in D3.2.



**Figure 14: Summary of TEACHING Platform Architecture**

The proposed architecture is a result of the collapsing of the layers mentioned in the previous section, implemented by the combined capabilities of the TEACHING Board and the suggested software toolkit. It is not possible to provide the full details of the implementation because it largely depends on the dependable system characteristics and the application domain. In what follows we provide recommendations for the creation of the APIs.

### AIaaS container (Application Runtime)

The AIaaS container or “Application Runtime”, is the heart of the software toolkit. It runs the processes for the deployment of AI models for training and inference. The Application Runtime implements all tasks that are common to support AI models and are common to all TEACHING

applications. The Application Runtime component manages the main execution workflow of an AIaaS application. Its core functions include:

- the instantiation of the underlying AI framework that hosts the learning modules;
- instantiation and execution of the AI models of the application;
- configuration of the AI data bus to correctly route the data stream to the relevant AI models.
- manage the application logic, triggering and providing data to its function.

This implies several key interoperation features that depend on the assumptions made on a specific implementation of the AIaaS support.

This flow is orchestrated through the use of the **Application Model** a YAML-formatted document that provides values to be set as environmental variables, effectively orchestrating data flows, ML pipelines and component interactions and synchronizations.

### Sensors API

The role of the Sensors API is to wrap the actual sensors that publish their streams to a message broker in a callable API that can serve data upon request from whichever module asks for the respective data. In order to implement this, the sensor API maintains a data buffer, which collects data as they arrive at the message broker. The buffer keeps the latest messages from each sensor. When a request to read data from the Sensor API is made the API returns the respective set of latest messages and automatically empties the respective queue in order to receive new messages from this sensor. This guarantees that the Sensor API provides the latest readings for a sensor every time it is called.

Every time a new sensor is added to the TEACHING platform an instance of the SensorAPI is instantiated in order to provide access to the sensor. The constructor (**init**) also defines the size of the buffer. The **open** method creates an instantiation of a connection to the message broker to the respective topic of each sensor.

The **read** method returns the content of the buffer and clears the buffer. Figure 8 shows the basic operations scheme of the Sensors API in the AIaaS system. In order to provide room for scalability, we assume that the drivers needed for each case are available in order for each sensor to be able to publish on the MQTT bus.

### Storage API

The aim of the Local Storage API is to provide the other TEACHING components with stored versions of the ML and AI models and also allow the long-term storage and reuse of newly trained models. Apart from models, the Storage API allows to store configurations, temporary files and caches, to store results and any other custom binary object.

For this purpose, the Local Storage API maintains an SQLite (SQLiteStorage class) for storing the various objects (instances of an Item class), which have an id, a name and description, a storage type, a timestamp and the filename where the actual object is stored. The API provides methods for adding items to the storage, retrieving them from the storage, getting their metadata by id or name and also for removing items from the storage. Finally, it has methods that allow

to store the item to the disk or retrieve it from the disk, but this functionality has not been used in the current mockup.

### External Services API

The proposed architecture is based on two principal technologies: MQTT<sup>5</sup> and Apache Kafka<sup>6</sup>. In our architecture, KAFKA components run partly on the mobile node and partly on the far edge. The Client component is used to develop the part of code that runs on the mobile node (Far Edge) and allows its connection with the infrastructure of KAFKA brokers; such brokers are implemented in the Near EDGE infrastructure. Precisely, a Consumer module has been developed to read the data published on the MQTT broker from the various sensor networks used onboard the mobile node or worn by the user. The Consumer, therefore, behaves as a subscriber of MQTT topics. Consistently exploiting the Client component, a Producer component has been developed to publish the messages of the MQTT topics on the KAFKA topics. Here the topics have only one partition that acts as a leader and is replicated a predetermined number of times on many brokers. These replications are the followers to give redundancy to the information. The Consumer module also reads the aggregated/federated model coming from the Cloud or the Near EDGE, or even from the Far Edge on behalf of the AI module, while the Producer module can publish this model in an MQTT topic if it is necessary to have it read by devices on board the vehicle that communicates through MQTT. Interoperability between KAFKA and MQTT is guaranteed through the KAFKA Connect component that, through the Sink and Source modules, can make KAFKA communicate with systems that do not speak KAFKA.

The KAFKA's component configuration is maintained on the Zookeeper platform located on the Near infrastructure or in the EDGE Cloud. Precisely in Zookeeper are maintained:

- The information is used by the KAFKA Controller, which is responsible for maintaining leader-follower relationships across all partitions. If a node for some reason goes down, it is the responsibility of the controller to tell all replicas to act as partition leaders in order to fulfil the duties of partition leaders on the node that is about to fail.
- Information about KAFKA brokers needed to elect a new Controller if the current one goes down. In Zookeeper it is maintained all information about active topics on KAFKA brokers, the number of partitions for each topic, the position of all replicas, the list of configuration overrides for all topics and which node is the favourite leader.
- The list of all brokers that are running at a given time and are part of the cluster.
- Access control lists (ACL) of all active topics on the brokers.

Note that the best practice is to exploit the redundancy of the configuration information provided by the Zookeeper repositories to avoid that the fault of an entity implies the fault of the whole infrastructure.

To allow the auto-configuration of a new node, the new node has been registered through a portal and download the configuration file, which contains the IP addresses of at least one KAFKA server, where the new node can request all the info such as topics, partitions cluster.

---

<sup>5</sup> MQTT: <https://mqtt.org>

<sup>6</sup> Kafka: <https://kafka.apache.org/>

The configuration procedure also allows the download of the certificate that the new node will use to authenticate itself with the broker. So, the infrastructure needs of a Certification Authority. KAFKA uses for its client-broker communication the TLS protocol. Once the node is connected, it receives info on the active brokers and topics, then new node and broker exchange their certificates. Once acquired, the respective certificates broker and new node communicate in an encrypted way. Note that the KAFKA ACLs, in our case, is used in such a way to avoid that each node can read the topics where its federated model is saved and cannot go to read the other ones.

## 8 Conclusions

WP1 is meant to provide technical strategy and oversight for all research and development activities throughout the lifecycle of the project. It also includes requirements collection and analysis, state of the art review, and architecture specification production. The activities within this work package strongly interact with those in the technical WPs providing specifications for the tools and software developed therein and taking care of their smooth integration. The specific objectives of this WP for M13-M20 were to:

- Define the TEACHING Platform architecture to feed it in the technical work packages
- Coordinate the efforts for creating an integrated, proof-of-concept platform

This report provides an overview of work conducted mainly in Task 1.2 and 1.3 during this period time towards the implementation of the first objective. The main outcomes presented here are the:

- Conceptual design of the TEACHING Platform based on the humanistic intelligence concept which revolves around four modules:
  - Monitoring of the state of the extended system that includes humans
  - Quantification of the human state
  - Integration with the dependable system (DMU)
  - Programming for the creation of new TEACHING Applications
- Non-functional and functional requirements analysis and definition of means of verification for each of them
- Key design decisions for the TEACHING Platform architecture
- Implementation recommendations for the TEACHING Platform

The remaining work involves the monitoring of the integration and implementation instances so as to evaluate NFRs against the defined KPIs and revise architecture as needed.



## 9 References

- [Kir08] R. Kirner and P. Puschner, “Obstacles in worst-case execution time analysis,” in 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), 2008, pp. 333–339.
- [Mez11] E. Mezzetti and T. Vardanega, On the industrial fitness of WCET analysis. na, 2011.
- [Bin14] J. Bin, S. Girbal, D. Gracia Pérez, A. Grasset, and A. Mériçot, “Studying co-running avionic real-time applications on multi-core COTS architectures,” Toulouse, France, Feb. 2014, Accessed: Dec. 17, 2020. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-02271379>.
- [Now12] J. Nowotsch and M. Paulitsch, “Leveraging Multi-core Computing Architectures in Avionics,” in 2012 Ninth European Dependable Computing Conference, May 2012, pp. 132–143, doi: 10.1109/EDCC.2012.27.
- [Wil08] R. Wilhelm *et al.*, “The worst-case execution-time problem-overview of methods and survey of tools,” *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, p. 36:1–36:53, May 2008, doi: 10.1145/1347375.1347389.
- [Hec05] R. Heckmann and C. Ferdinand, “Verifying safety-critical timing and memory-usage properties of embedded software by abstract interpretation,” in *Design, Automation and Test in Europe*, Mar. 2005, pp. 618–619 Vol. 1, doi: 10.1109/DATE.2005.326.
- [Sat] R. Sattiraju, D. Wang, A. Weinand, and H. D. Schotten, “Link level performance comparison of c-v2x and its-g5 for vehicular channel models,” in 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring). IEEE, 2020, pp. 1–7.
- [Mol] A. Moller, J. Nuckelt, D. M. Rose, and T. Kurner, “Physical layer performance comparison of lte and ieee 802.11p for vehicular communication in an urban nlos scenario,” in 2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall). IEEE, 2014, pp. 1–5.
- [Mar] V. Marojevic, “C-v2x security requirements and procedures: Survey and research directions,” arXiv preprint arXiv:1807.09338, 2018.
- [Ko19] C.-Y. Ko, Z. Lyu, L. Weng, L. Daniel, N. Wong, and D. Lin, “Popqorn: Quantifying robustness of recurrent neural networks,” in International Conference on Machine Learning. PMLR, 2019, pp. 3468–347.
- [Aba16] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, Li Zhang, Deep Learning with Differential Privacy, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, October 2016
- [Dimi20] Dimitrakopoulos, G., Uden, L., Varlamis, I.: The future of intelligent transport systems. Elsevier, Amsterdam (2020)
- [Mol18] Molnar, L.J., Ryan, L.H., Pradhan, A.K., Eby, D.W., St. Louis, R.M., Zakrajsek, J.S.: Understanding trust and acceptance of automated vehicles: An exploratory simulator study of transfer of control between automated and manual driving. *Transportation Research Part F: Traffic Psychology and Behaviour* 58, 319–328 (2018).
- [Cle21] P. Clement, H. Danzinger, O. Veledar, C. Koenczoel, G. Macher and A. Eichberger, “Measuring trust in automated driving using a multi-level approach to human factors,” in In Print (Euromicro DSD/SEAA Conference 2021), Palermo, 2021.
- [Mal17] Malhotra, P., TV, V., Vig, L., Agarwal, P., & Shroff, G. (2017). TimeNet: Pre-trained deep recurrent neural network for time series classification. arXiv preprint arXiv:1706.08838.
- [Pang20] Pang, G., Hengel, A. V. D., Shen, C., & Cao, L. (2020). Deep reinforcement learning for unknown anomaly detection. arXiv preprint arXiv:2009.06847.