Horizon 2020 Program (2014-2020)

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

D1.1: Report on TEACHING related technologies SoA and derived CPSoS requirements[†]

Contractual Date of Delivery	31/10/2020
Actual Date of Delivery	30/12/2020
Deliverable Security Class	Public
Editor	Konstantinos Tserpes (HUA)
Contributors	UNIPI: Davide Bacciu, Daniele Mazzei, Gabriele Mencagli HUA: Konstantinos Tserpes, Dimitrios Michail, Iraklis Varlamis, Charalampos Davalas, Christos Sardianos, Florios Glezos CNR: Patrizio Dazzi TUG: Georg Macher, Maid Dzambic AVL: Herbert Danzinger, Philipp Clement, Omar Veledar I&M: Lorenzo Giraudi, Roberta Peroglio M: Calogero Calandra, Marilina De Gennaro TRT: Sylvain Girbal, Jimmy Le Rhun ITML: Siranush Akarmazyan IFAG: Jakob Valtl
Quality Assurance	Stefano Chessa (UNIPI)

[†] 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. Stefano Chessa, (UNIPI)

Revisions

Version	Date	By	Overview
1.0	30/12/2020	Coordinator	Final
0.9	29/12/2020	Editor	Addressed Comments
0.8_	22/12/2020	Reviewer	Review comments
0.8	20/12/2020	Editor	Review version
0.1	16/11/2020	Editor	ToC

Table of Contents

List of Tables

List of Figures

List of Abbreviations

EC	European Commission
WP	Work Package
DoA	Description of Action
CPS	Cyber-Physical System
SoS	System Of Systems
NFR	Non-functional Requirements
ML	Machine Learning
DL	Deep Learning
ADAS	Advanced Driver-Assistance System
FMS	Flight Management System
MIAC	Model Identification Adaptive Control
MPC	Model Predictive Control
PoT	Proof-of-Trust
TSTP	Trustful Space-Time Protocol
HLA	High-level architecture
SOTIF	Security of the intended functionality
TSC	Technical Safety Concept
DAL	Design Assurance Level
WCET	Worst Case Execution Time
COTS	Commercial off-the-shelf
HCD	Human-centered design
DPS	Deterministic platform software
SoC	System on Chip
SoM	System On Module
PEC	Physiological, emotive, and cognitive

Executive Summary

The objective of this deliverable is to delineate the TEACHING CPSoS and define the scope of the research and development work of the TEACHING project. It focuses on the definition of the high-level concepts thus guiding the more technical-oriented work packages. The report provides a general, theoretic definition of the TEACHING CPSoS and the applications that it can support. It elicits the requirements and resorts to the literature for the identification and analysis of the relevant research and technical challenges. This analysis is then used for the definition of the baseline tools and technologies. The latter is achieved through conducting a survey of existing and well-established platforms. This output was important so as to allow the rest of the work packages to delve into the details of their tasks. Finally, in order to further facilitate the bridging of the TEACHING CPSoS with the more technical work, we provide a conceptual view of its architecture.

1 Introduction

This document provides a thorough account of the work and the results followed during the course of *Task 1.1: Requirement identification of the TEACHING project* (T1.1). This Task was active during the period 01/01/2020-31/12/2020 and the work was conducted by the following partners: HUA (leader), TUG, AVL, MM, I&M, TRT, IFAG.

This work was complemented with the work in *Task 1.3: Human-centered design* (T1.3). This task was active durine the period 01/01/2020-31/12/2020 and the work was conducted by the following partners: UNIPI (leader), CNR and IFAG

The objectives of the work in T1.1 were to:

- record the requirements of the TEACHING CPSoS
- provide a detailed analysis of the state of the art aimed at tracking the technological trends in the scope of the TEACHING project
- analyse the market of CPS with the objective of identifying suitable hardware platforms, sensors kits and technologies and available software tools.

whereas the objective of T1.3 was to study the interfacing between the TEACHING technology and systems and the users, thus, becoming a subset of the overall T1.1 objectives that emphasized on Human Centered Design (HCD).

Bringing about those objectives provides a guidance to the Research and Development (R&D) work of the project in its effort to deliver its promised outcomes. Those outcomes are defined in 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 three main artefacts:

- A computing platfom
- A development and deployment software toolkit
- Autonomous, adaptive and dependable CPSoS applications

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 **CPSoS applications**.

In order to bring about the T1.1 objectives, the participating partners worked in close collaboration with work packages (WP) 2-5 in order to understand what are the use cases of interest, the relevant technology trends as well as engineering methodologies.

All those WPs provide a view of the project requirements deriving from a different perspective. WP2-4 aim mainly into providing the technical requirements and WP5 focus on the definition of the tangible use case requirements. WP1 focused on the definition of the TEACHING CPSoS **(system) requirements** that are linked with its implementation. Hence, the analysis conducted in this work, remains at a conceptual level and its main aim was to provide a guidance to the rest of the WPs that delve into the details of those concepts.

To convey the results of this work, this report starts with a theoretical definition of the TEACHING CPSoS (Section 2) and then delves into the details of the TEACHING CPSoS

Applications (Section 3). The latter is a central step in the identification of the requirements and the definition of the system actors and their roles. Focusing on the identified non functional requirements (NFR), the work proceeded with analysing the relevant state of the art approaches so as to better understand the research and technical challenges for the implementation of the TEACHING CPSoS (Section 4). Subsequently, the analysis went on to the hardware and software tools that are currently available that could meet the NFRs and as such, could form a baseline for the TEACHING Platform (Section 5). Leveraging on the functional and non-function requirements, the approach move on a step further and started working in the definition of the TEACHING CPSoS conceptual architecture. This is of course, a subject of another deliverable (TEACHING Report *D1.2: TEACHING CPSoS architecture and specifications*), however, a preliminary version is already presented in Section 6.

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	TEACHING CPSoS	Provide an umbrella definition of the TEACHING CPSoS
Section 3	CPSoS applications and system requirements	Identify the functional and non-functional requirements and the key actors of the TEACHING CPSoS
Section 4	Research and technical challenges	Identify the research and technical challenges for meeting the NFRs defined previously in a CPS context
Section 5	Baseline technologies and tools	Examine the existing, COTS tools and technologies upon which TEACHING could rely in order to meet the identified requirements
Section 6	TEACHING Platform architecture	Leverage the identified requirements to provide a preliminary version of the TEACHING Platform architecture
Section 7	Conclusions	Summarize findings and present future work

1.1 Relationship with other deliverables

There is a group of related deliverables, i.e. D1.1, D2.1, D3.1, D4.1 and D5.1 (Table 2), all of which serve as a mean of milestone MS1 verification. That is the first project milestone, entitled *Release of the TEACHING design (requirements, specification and architecture)*.

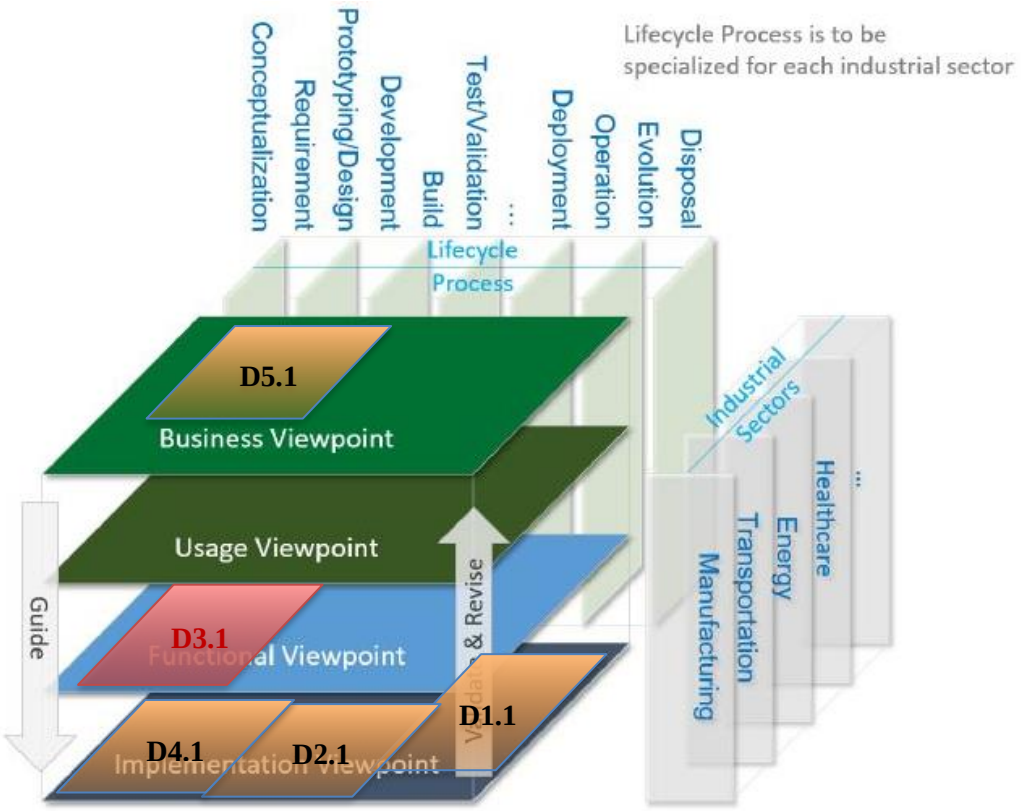

Figure 1: Depiction of the IIRA Viewpoints from [1] and mapping of focus of TEACHING Deliverables MS1

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.

Table 2: Deliverable grouping for verification of TEACHING Milestone 1

D1.1	Report on TEACHING related technologies SoA and derived CPSoS requirements
D2.1	State-of-the-art analysis and preliminary requirement specifications for the computing and communication platform
D3.1	Initial Report on Engineering Methods and Architecture Patterns of Dependable CPSoS
D4.1	Initial report on the AIaaS system
D5.1	Initial use case specifications

[1] https://iiot-world.com/industrial-iot/connected-industry/iic-industrial-iot-reference-architecture/

2 TEACHING CPSoS

In order to be able to define the TEACHING CPSoS, we need to establish a commonly accepted definition of Cyber-Physical Systems. The literature includes a large number of such definitions yet there are a few that accurately express the TEACHING concepts. Such is the one from Platzer [1] who proposes the following:

Cyber-physical systems combine cyber capabilities with physical capabilities to solve problems that neither part could solve alone. Cars, aircraft, and robots are prime examples, because they move physically in space in a way that is determined by discrete computerized control algorithms that adjust the actuators (e.g., brakes) based on sensor readings of the physical state.

Similarly, the Ptolemy Project website[2], maintained by the EECS department at Bekeley, states:

CPS integrates the dynamics of the physical processes with those of the software and networking, providing abstractions and modeling, design, and analysis techniques for the integrated whole. Computer science, as rooted in the Turing-Church notion of computability, abstracts away core physical properties, particularly the passage of time, that are required to include the dynamics of the physical world in the domain of discourse.

We employ these definitions in order to define the TEACHING CPSoS. The rationale is developed in what follows.

Assume a cyber-physical system (CPS) whose behavior is regulated by a feedback control loop. This system can be a vehicle or an airplane and its behavior ranges from simple navigation to emergency maneuvers. This system is stable/dependable in the sense that it is achieves its objectives which are dictated by concrete policies, under a wide range of conditions. TEACHING introduces a new objective to the control loop, through the integration of a new, potentially undependable 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.

[2] https://ptolemy.berkeley.edu/index.htm

3 CPSoS applications and system requirements

A number of partners in the project consortium represent the TEACHING CPSoS end users. Those partners have already provided a description of two use cases that once implemented, will add value to their business endeavors. Analyzing the use cases and scenarios that revolve around the use of the TEACHING CPSoS is a typical way to elicit system requirements.

The use cases revolve around the domains of autonomous navigation of vehicles and aircrafts. Their details along with some domain-specific use case scenarios can be found in the TEACHING Report *D5.1: Initial use case specifications*. The particular user requirements are listed in the Project requirements document release 1.0 in the form of a jointly edited, living document for tracking and evaluating progress but also to facilitate change control. In T1.1 we depart from the definitions provided in D5.1 and work on a higher level of abstraction since we are interested to define the TEACHING CPSoS applications. The implementation of those applications will serve as a proof of concept for the underlying technology that TEACHING is developing, i.e. the TEACHING CPSoS.

Towards that end, we start with the identification of the TEACHING CPSoS actors and their roles (Section 3.1) before we provide the overview of those CPSoS applications and the associated high-level functional and non-functional requirements for the TEACHING CPSoS (Sections 3.2 and 3.3).

3.1 Actors

We employ the TEACHING goal statement that introduces the concept of a platform for development and deployment. The latter implies that there is an owner and a user of the platform. Furthermore, we distinguish between the actors with a business-oriented role and their agents, i.e. actors with a technical role.

3.1.1 Business roles

3.1.1.1 Platform Provider

The Platform Provider refers to the entity that owns and maintains the TEACHING Platform. This entity must have a business incentive to do so. Such incentive may be domain-specific, e.g. a vendor installing the TEACHING Platform along with a TEACHING application in vehicles. An alternative option would be for the vendor to provide a general-purpose TEACHING Platform allowing other business entities to develop and deploy their applications on top of it. Such a business entity is referred to as “application provider”.

3.1.1.2 Application Provider

The Application Provider is a business entity that owns an application that can host an instance of the TEACHING Platform. This actor possible pays a fee to the Platform Provider in return for the use of the Platform.

3.1.2 Agents

3.1.2.1 Platform administrator

This is the Platform Provider’s agent, i.e. a person or group of people that perform operations related to the configuration, upkeep and reliable operation of the TEACHING Platform.

3.1.2.2 Application Developer

This is the Application Provider's agent, i.e. a person or group of people that can deploy and possibly develop the application in the TEACHING Platform.

3.2 Functional requirements

In this section we introduce the two use cases in order to identify the high-level system functional requirements. To achieve that, we parse the narrative pertaining to the main scenarios of the use cases.

3.2.1 Use case #1: Automotive

The developer wants to implement an application that performs online learning based on human feedback so as to feed a vehicle's Advanced Driver-Assistance System (ADAS) and have it adapt to the comfort levels of the passengers (Figure 2). 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.

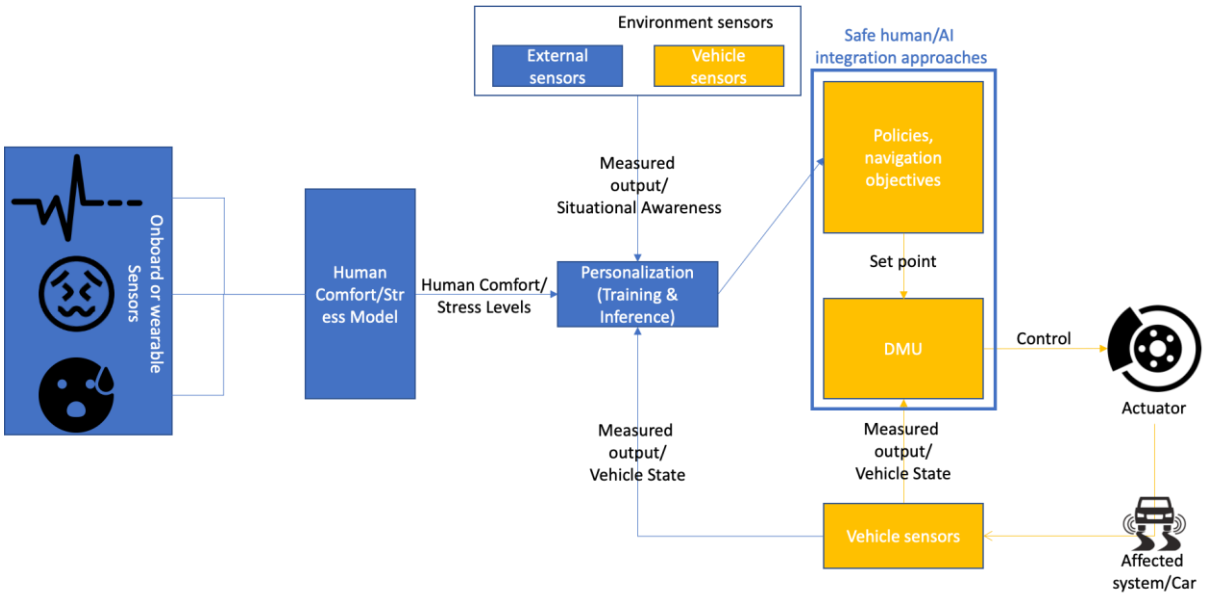

Figure 2: Automotive CPSoS Application

In such an application we recognize the following **high-level functional requirements**: A computing platform and software toolkit that integrates with the onboard ADAS and interacts with it (FR1.1). This system must be able to communicate with the wearable sensors (FR1.2) and the web (FR1.3). The system must allow the execution of software directly or in the form of container images (FR1.4). It should also be able to execute machine/deep learning algorithms for training (FR1.5) or inference (FR1.6). Finally, it must be able to offload tasks at the edge nodes and generally enable an interplay between local and remote resources (FR1.7).

3.2.2 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 3). 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.

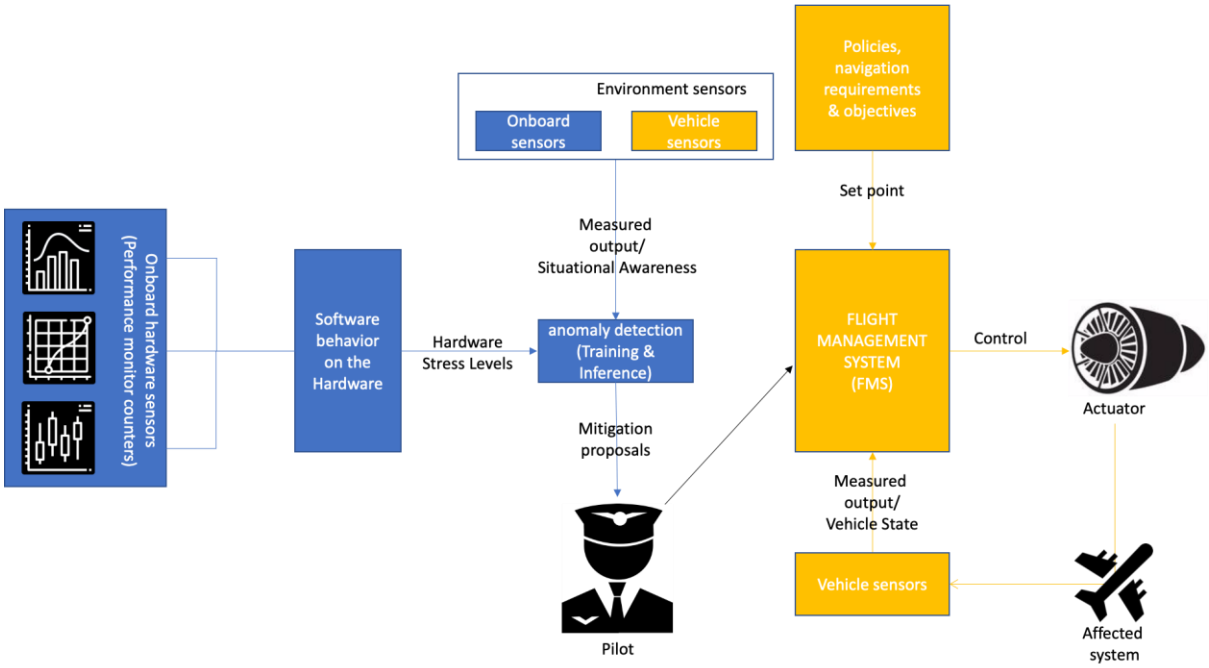

Figure 3: Avionics CPSoS Application

In such an application we recognize the following **high-level functional requirements**: A computing platform and software toolkit that integrates with the onboard FMS and retrieve data from its monitoring probes (FR2.1). The system must allow the execution of software directly or in the form of container images (FR2.2). It should also be able to execute machine/deep learning algorithms for training (FR2.3) or inference (FR2.4). It must also be able to communicate the results of its processing to the human pilot (FR2.5). Finally, it must be able to offload tasks at the edge nodes and generally enable an interplay between local and remote resources (FR2.6).

3.3 Non Functional Requirements

Requirements' analysis dictates the identification of the non-functional requirements (NFR). Unlike their functional equivalents, it is safer to elicit NFRs through a more general analysis rather than the parsing of the use case scenarios. It is often the case that NFRs are considered to be "obvious" and may escape the use case scenarios analysis. In many cases, however, there is a fixed list of NFRs that pertain to the development of a system. The system designers and developers select a subset of those, based on granularity level of the analysis and the desired emphasis of the system.

Following this rationale, we resort to the identification of a specific number of NFRs as those are explicitly stated in the scope of the project as phrased in the Description of Action (ref Grant Agreement Appendix). Of course, the number of NFRs applied to CPSs is quite extended and they are conveniently illustrated in the CPS concept map presented in Figure 4.

Notwithstanding, the definitions and hierarchy of each NFR varies depending on the viewpoint allowing much room for debate.

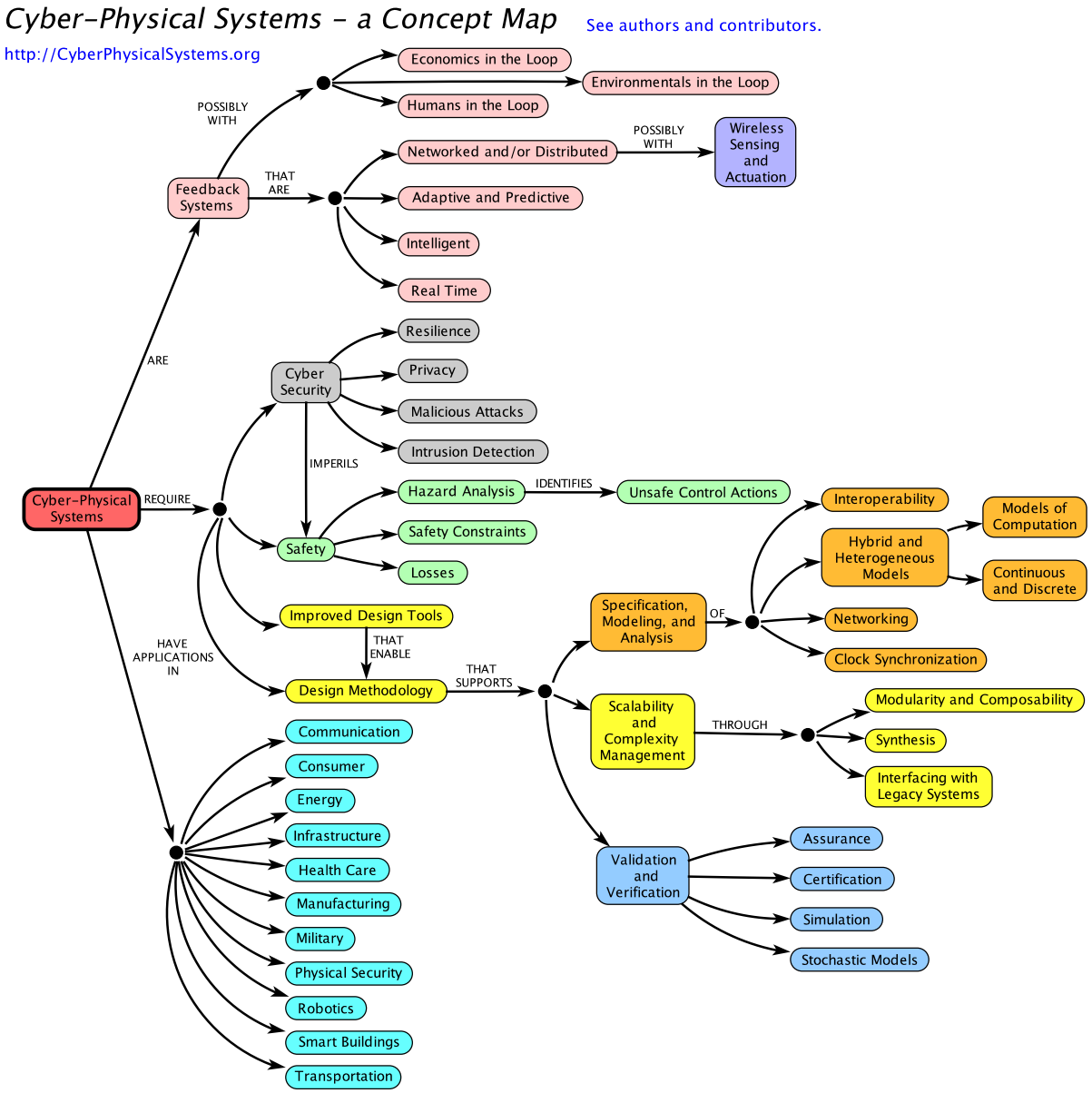

Figure 4: CPS concept map (source: Ptolemy Project website[3])

Among the superset of properties that define a CPS as presented in Figure 4, TEACHING focuses on: adaptability (NFR1), security (NFR2), dependability (decomposed to safety and reliability) (NFR3), privacy (NFR4), acceptability (NFR5) and energy efficiency (NFR6). Those are better explained in what follows.

- **Adaptability**: CPS systems are typically closed-loop systems, where sensors make measurements of physical processes, the measurements are processed in the cyber subsystems, which then drive actuators that affect the physical processes. The control strategies implemented in the cyber subsystems need to be adaptive (responding to changing conditions) and predictive (anticipating changes in the physical processes).

[3] https://ptolemy.berkeley.edu/projects/cps/

- **Security**: The overarching goal of cybersecurity is to build trust in systems.
- **Dependability.Safety**: The goal of safety is to assure that the system will not misbehave in a manner that transitions the system to hazardous states and, therefore, is susceptible to causing losses in general and accidents in particular.
- **Dependability.Reliability**: In the context of CPS, reliability is the ability of a system to continue operating satisfactorily when stressed by unexpected inputs, subsystem failures, or environmental conditions or inputs that are outside the specified operating range. Fault tolerance, fault detection, and adaptation are all techniques the promote resilience.
- **Privacy**: In the context of CPS, privacy is the problem of protecting information about humans from unauthorized access by other humans or machines.
- **Acceptability (human-centric design perspective)**: Many cyber-physical systems include humans as an integral component. Humans are very difficult to model, so understanding and validating such systems becomes particularly challenging.
- **Energy Efficiency**: Most of the embedded platforms that perform inference have stringent energy consumption, compute and memory cost limitations; efficient processing has thus become of prime importance under these constraints.

The reason for emphasizing on those NFRs originates to the initial analysis on the challenges that the TEACHING project was set to tackle. Given this, T1.1 along with T1.3 focus on the analysis of the relevant non-functional requirements (NFR) of the CPSoS. This analysis included the review of the technologies and tools that are considered to be the state-of-the-art in CPS when they are to deal with the identified NFRs. This review is presented in the following Section whereas Section 5 presents the platforms that are meant to meet those requirements.

4 Research and technological challenges

In what follows we provide an account of the R&D challenges that we need to deal with in order to meet the NFRs identified as important for the TEACHING CPSoS in Section 3.3. For that reason, we conducted an analysis of the state of the art for each of the NFRs. This literature survey was important in order for the project partners to update their knowledge bases, communicate their domain knowledge to the consortium and establish the research foundation upon which TEACHING will build.

4.1 Adaptability

In principle, the TEACHING CPSoS concept aims at augmenting the context that the initial control loop-based system is trying to control by adding more objectives to the controller and a new array of sensors to assess its performance[4]. Essentially, the result of the integration of the dependable/undependable systems results in a new system that bears an optimized controller. The key characteristic of this new system is that the introduction of the new objectives comes with a great deal of uncertainty. The presence of the human factor is anticipated as the main source of such uncertainty. The range of the required operations of the new system in order to meet human-related objectives varies greatly.

This observation leads us to seek guidance for the design of this kind of systems to the domain of Adaptive Control.

Definition[5]: Adaptive control is the control method used by a controller which must adapt to a controlled system with parameters which vary or are initially uncertain. For example, as an aircraft flies, its mass will slowly decrease as a result of fuel consumption; a control law is needed that adapts itself to such changing conditions. Adaptive control is different from robust control in that it does not need a priori information about the bounds on these uncertain or time-varying parameters; robust control guarantees that if the changes are within given bounds the control law need not be changed, while adaptive control is concerned with control law changing itself.

Even though control engineering as well as feedback found in nature are not targeting software systems, mining the rich experiences of these fields and applying principles and findings to software-intensive adaptive systems is a most worthwhile and promising avenue of research for self-adaptive systems [2].

Systems enabled with self-adaptive capabilities continuously sense their environment, analyze the need for changing the way they operate, as well as plan, execute and verify adaptation strategies fully or semi automatically. On the one hand, the goal of software evolution activities is to extend the lifespan of software systems by modifying them as demanded by changing real-world situations. On the other hand, control-based mechanisms, enabled through self-adaptation, provide the means to implement these modifications dynamically and reliably while the system executes [3].

Hence the question is to determine the equations that govern the dynamic behavior of the new system.

[4] Another way to view the problem, is that in a system with multiple, concurrent control loops (Multiple Input Multiple Output-MIMO), we need to add one more, but one that it is not easy to model. At the same time, we need to change the overall SoS objective function.

[5] https://en.wikipedia.org/wiki/Adaptive_control

4.1.1 System identification

In control theory the process of determining the model/equations that govern the dynamic behavior of the new system is referred to as "**system identification**" and a well-established reference model for it is the Model Identification Adaptive Control (MIAC).

In MIAC (Figure 5), the reference model that allows parameter estimation is identified or inferred at runtime using system identification methods, i.e., using the control input and measured output to identify the reference model. Then, the new model parameters are calculated and sent to the adjustment mechanism which calculates the parameters that will modify the controller.

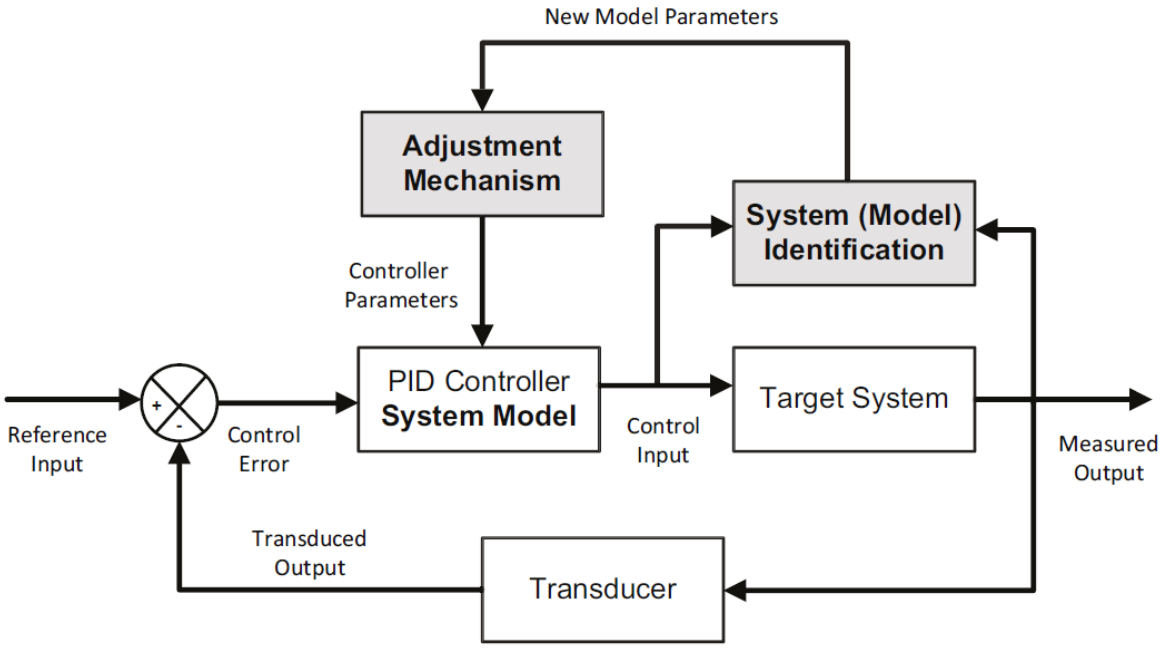

Figure 5: Model Identification Adaptive Control (MIAC)

Note that the notion of "model" in control theory adheres to the definition of "white box model" that is commonly used in the field of computational intelligence. Based on the latter, process control methods were proposed and commonly used with white box models.

White box models describe a system from first principles, e.g., a model for a physical process that consists of Newton equations. However, in cases such as the one we are studying, such models are overly complicated or even impossible to obtain due to the complex nature of many systems and processes (natural or artificial). A much more common approach is therefore to start from partial knowledge of the behavior of the system and its external influences (inputs), and try to determine a mathematical relation between inputs and outputs without going into the details of what is actually happening inside the system [4].

TEACHING considers the MIAC reference model, extended by the use of ML approaches to approximate the reference model (or conduct the system identification in the control theory terminology) as an appropriate approach for its purposes.

4.1.2 Model Predictive Control

Another option to be considered is Model Predictive Control (MPC). In its basic form, MPC (Figure 6) consists of the following two steps executed repeatedly on-line. First solve an

optimization problem to determine over a finite time horizon the open-loop optimal control input trajectory with respect to a certain performance criterion. Then apply the first part of it to the system to be controlled and repeat the whole procedure to achieve a desired closed-loop behavior.

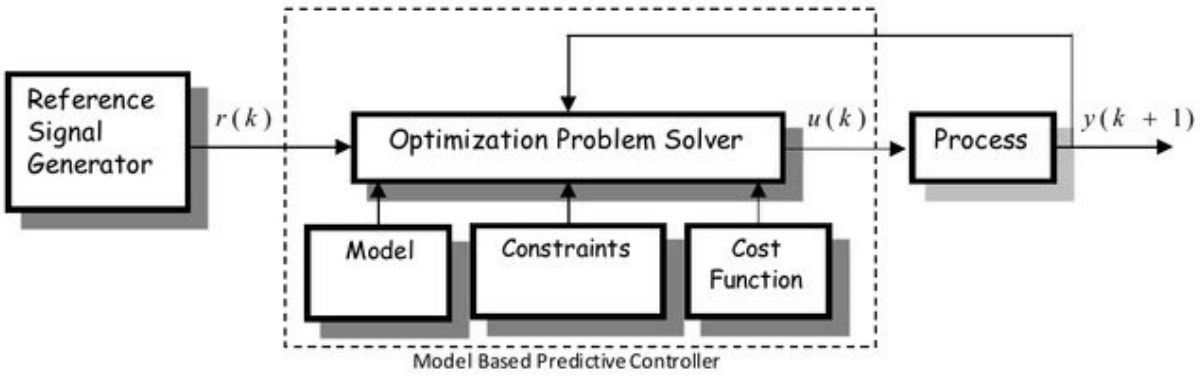

Figure 6: Model-based Predictive Controller diagram [Source: [5]].

The numerical effort of solving this optimization problem on-line can be prohibitive in fast or large scale applications. Hence, a large number of results aiming at alleviating this obstacle has been presented in the literature. Therein, the online optimization task is simplified or even (partly) avoided by shifting some of the computational effort off-line. Some of these results are based on machine learning techniques.[6]

What is of interest, is that MPC seems an appropriate choice for Multiple Input Multiple Output (MIMO) control systems. TEACHING focuses on such systems, where multiple controllers are manipulating the variables of different systems/processes, each of which provides a measurable output [6].

The premise upon TEACHING would be building in this case, is that it can approximate adaptive MPC control with reinforcement learning algorithms while taking all appropriate measures to deal with stability, feasibility, robustness, and constraint handling. Table 3 presents how MPC and Reinforcement Learning can complement one another.

Table 3: Properties of Model Predictive Control and Reinforcement Learning (Source: [7])

Property	Model Predictive Control	Reinforcement Learning
Model	required ✗	not required ✓
Convexity	required (usually) ✗	not required ✓
Adaptivity	immature (usually based on robustness) ✗	mature (inherent) ✓
Online Complexity	high (except explicit and neural MPC) ✗	low ✓
Offline Complexity	low (except explicit and neural MPC) ✓	high ✗
Stability Theory	mature (e.g. based on terminal cost) ✓	immature ✗
Feasibility Theory	mature (e.g. based on terminal constraints) ✓	immature ✗
Robustness Theory	mature (e.g. based on tubes or ISS) ✓	immature ✗
Constraint Handling	mature (inherent) ✓	immature (except input constraints) ✗

4.2 Dependability and Security

For making dependability more measurable, Aviziensis et al. [8] define dependability as the ability of a system to avoid service failures that are more frequent or more severe than acceptable. This definition has overlaps with that of Security, leading in an intertwined

[6] https://ipvs.informatik.uni-stuttgart.de/mlr/colloquia/dipl-ing-gregor-goebel-simplifying-model-predictive-control-algorithms-via-machine-learning-techniques/

perspective when we refer to CPS systems. Figure 7shows the overview of the attributes, threats and means to attain dependability and security.

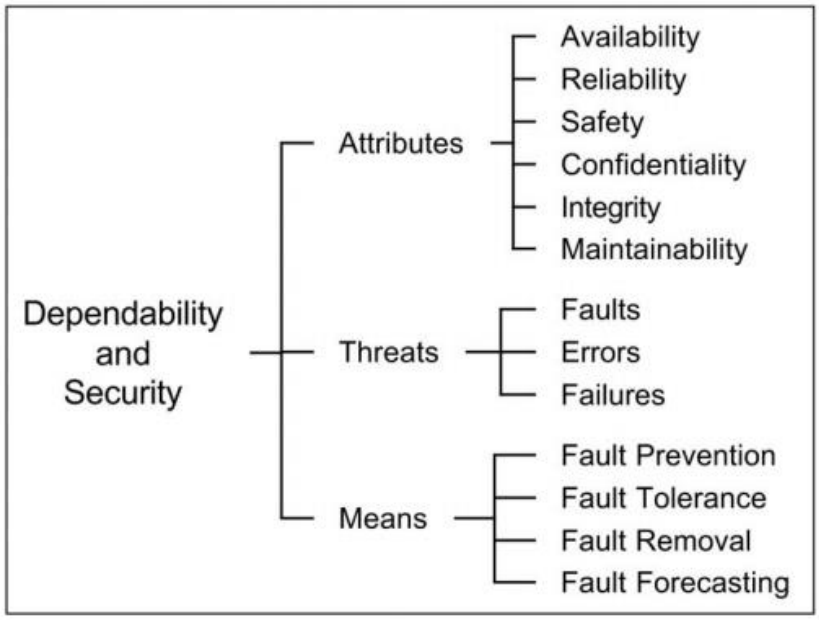

Figure 7: Attributes, threats, and means to attain dependability and security

Based on [8], the dependability and security attributes can be summarized as:

- **Security.Integrity** - "absence of improper system alterations. Includes both intentional and unintentional interference in the system.
- **Security.Availability** - "readiness for correct service. This attribute is often expressed as a function of time and represents the probability of correct service at a given time.
- **Security.Confidentiality** - the security principle that controls access to information. It is designed to ensure the wrong people cannot gain access to sensitive information while ensuring the right people can access it.
- **Dependability.Maintainability** - ability to undergo modifications and repairs. Maintainability indicates, how easy it is to restore a system which provides incorrect service to provide correct service again.
- **Dependability.Reliability** - "continuity of correct service. This attribute represents the probability of a correct service being delivered during a specific time interval.
- **Dependability.Safety** - "absence of catastrophic consequences on the user(s) and the environment.

In what follows we analyze each of these properties in two main categories: security and dependability.

4.3 Security

Cyber-Physical systems, like any cyber system, are prone to security threats; such security threats can be the cause of damages to critical infrastructure, even the loss of human lives. Cyberattacks can happen on all layers of CPSs resulting in a multitude of potential threat models. Such threat include those directed to nodes which involve sensors and actuators threats causing data leakage, damage and security issues during massive data integration or loss of user privacy, incorrect access control policies and inadequate security standards [9].

There is, however, a plethora of solutions which are being developed to overcome such threats, since assuring cyber physical systems' security has been an issue of utmost importance lately. The main issue concerning CPS security threats is that it is harder to detect and stop them compared to regular internet attacks. This is due to their heterogeneous nature, their reliance on private and sensitive data, performance of involved devices, their large-scale deployment due as well as the prolonged procedures many hackers follow. Another problem which makes CPS's security harder to develop is that many real-life CPS are not open to scientific security search; in turn this fact delays the process of enforcing the security of such systems and makes it harder to detect possible breaches in the foreseeable future.

In Figure 8, six (6) main security objectives are presented. Among them integrity, availability and confidentiality are utmost important properties of computer security and still applicable requirements also for the security of a CPS. In fact, CPS needs strong integrity and availability of infrastructure and information, especially considering the fact that CPS systems are deployed in remote and harsh environments and involve interactions between the massive entities which may span heterogeneous wireless networks. In what follows we provide a comprehensive overview on these three (3) security-relevant properties and as well as approaches to ensure these objectives.

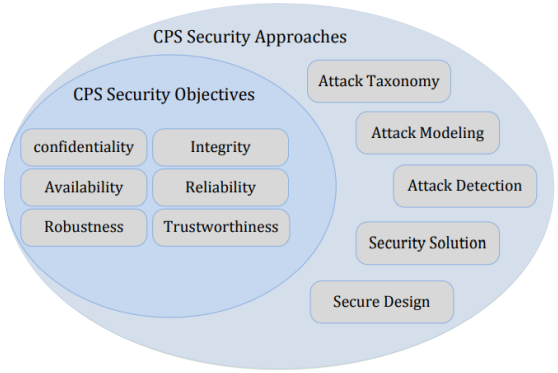

Figure 8: The Framework of Cyber Physical System Security [10]

4.3.1 Integrity

Integrity in the security of CPSs means that the system/device could be modified cannot be modified without authorization. Ensuring integrity in cyber physical systems is of great essence to the overall security of the system with regard to safeguarding the system's correct and reliably operation. To have a high comprehensive level of integrity protection in a CPS, one should involve several axes; such axes include the component level of the system, which contains sensors/actuator devices, up to control and supervisory systems, planning and configuration management, and the system life cycle. In this way, one can detect integrity violations in a system level in a reliable manner through analyzing integrity measurements from many unique integrity sensors, which capture and analyze integrity measurement from the physical world, on the field level, and of control and supervisory systems [11]. The challenge to detecting integrity security breaches is through detecting any change of information inside the CPS, where the information changed can be with malicious or non-malicious intent. It is, however, important for a CPS to detect and notify when such changes occur, so the user would know that a possible attack is going to occur or occurring [12].

Many works in the scientific literature address integrity security breaches and how to avoid them [11], [13], [14]. Among the solutions, we mention SCADMAN[15], a system that

preserves the Control Behavior Integrity (CBI) of distributed cyberphysical systems. In this system, the effectiveness of the controllers is verified by observing the wide system's behavior. This allows SCADMAN to detect a wide range of attacks against controllers, like programmable logic controller (PLCs), including malware attacks, code-reuse and data-only attacks. SCADAMAN's developers implement and evaluate the system based on a real-world water treatment testbed for research and training on ICS security. The test resulting from the deployment of SCADMAN show that it can detect a wide range of attacks including those that have previously been undetectable by typical state estimation techniques. Another team from the University of Santa Catarina in Brazil developed an architecture that protects the data integrity of any IoT-device in a system, including cyber physical systems, by utilizing blockchain features; their system is comprised of many layers, each one matched with their resource of data. To elaborate, the first layer of the system is comprised of sensors, gateways, and actuators; when combined together, they introduce the concept of Proof-of-Trust (PoT) which is an energy efficient, time-deterministic, and secure correspondence based on the Trustful Space-Time Protocol (TSTP). The upper levels are accountable for keeping information perseverance and integrity confirmation in semi-confided storage. The work additionally contains a performance assessment of a critical way of data to show that the design respect time-bounded activities requested by the sense-decide-actuate cycle of CPS.

4.3.2 Availability

Availability in a cyber physical system aims to always provide service even during system computing, control, and communication corruptions. High availability CPS needs the following characteristics: the physical layer can automatically deal with hardware failure, system updates, power load and other emergencies, to provide the correct service; information layer to deal with denial of service attacks, to provide the right information processing services. Availability malfunctions might happen due to a plethora of reasons including hardware and software failures, power cuts, system upgrade, and cyber-attacks, especially denial-of-service attacks. The system should strive at providing the required level of availability during a malfunction by maintaining redundant systems. An example of the importance of availability in a cyber physical system is in medical applications, that physicians acquire certain data in order to save a patient's life. If the availability of the data is compromised due to a cyber-attack, the physicians may not be able to take the necessary measures to save lives [16]. S. Parvin et al. [17] have worked on the enhancement of availability of CPS and proposed a multi-cyber (computational unit) framework to improve the availability of CPS based on Markov model. They evaluated the effectiveness of the proposed framework in terms of availability via offering multiple cybers. In this way, the whole system is available even though other units are not functioning. Sanislav et al.[18] mentioned different research challenges to achieve dependability in cyber-physical hydropower systems (CPHS). They revealed that the system's availability should be ensured with a view to achieving CPHS dependability.

4.3.3 Confidentiality

Confidentiality means that cyber physical systems should preserve authorized restrictions on information access, prevent the disclosure to unauthorized individuals or systems and protect personal privacy and proprietary information. The system should not allow the disclosure of any unauthorized personnel to the CPS; a viable way to ensure confidentiality is to investigate the problem of scheduling periodic messages with both time-critical and security-critical requirements and build a risk-based security profit model measuring the security quality of messages, trying to incorporate confidentiality improvement into message scheduling which expose critical messages to security threats, especially by confidentiality attacks [10]. An

example of confidentiality is when the medical reports of a certain patient are transported from the public health records system to a certain doctor or a clinic's system; the system needs to have confidentiality through the encryption of the records by limiting the places where the reports show and by the restricting the access of people for the stored reports [11]. If there was any breach by persons who are not authorized to accessing the reports, there is a confidentiality breach.

A way to protect a CPS from possible confidentiality breaches is through protecting the information between controllers, sensors and actuators from eavesdropping, which makes the assurance of confidentiality not enough if not combined with the prevention of infiltration by outsiders. To address the security vulnerability of confidential violation, a group of researchers develop a basis for a CPS security model by composing simple building blocks into a more complex system, and then examine the information security specifically geared towards preserving the event confidentiality in CPS [10]. Furthermore, Laura Vegh et al.[19] has presented a solution for ensuring data confidentiality and security by combining some of the most common methods in the area of security – cryptography and steganography. Furthermore, they have used hierarchical access to information to ensure confidentiality and also increase the overall security of the cyber-physical system with robust, reliable and flexible features. A cryptographic service was also used by W. Jiang et al. [20] to implement confidentiality protection for messages delivered over distributed CPSs and deploy fault detection within confidential algorithm to resist fault injection attacks.

Finally, hardware/software co-design techniques were also leveraged to accelerate confidentiality protection [21].

4.3.4 CPS Security Approaches

Compared to Internet attacks, attacks on CPS are more difficult to detect and prevent. To evade detection, hacks may apply multiple attack stages to gain the access to a CPS. The figure below illustrates the tree diagram of various attacks and threats on CPSs.

Figure 9. Attacks and Threats on CPSs

In the open literature there have been observed several security approaches to address attacks on cyber physical system. The possible security attack on the CPS and the way to prevent it is shown in the table below [22]. Among the listed attacks the denial-of-service attack can be quite dangerous in the smart-car industry; an attacker can simply cause damage by stopping the car windows from closing after being open, they can also cause damage to the Anti-lock braking system (ABS) and stop the car from stopping altogether.

Type of attack	**Security methods**
Eavesdropping	Cryptosystem (symmetric and asymmetric) secure routing and anonymous routing.
Compromised-key attack	Cryptography, key transport protocol, key agreement protocols, and two-party key establishment protocols.
Man-in-the-middle attack	Message digest, digital signature, MAC, biometrics, and trusted platform module
Denial of service attack	network traffic monitoring, analysis and filtering, antispoofing, DoS source traceback, etc.

Using Attack modeling techniques, the design of CPS can be planned ahead to ensure that the compromised components of the system are being protected from possible threats [23]. Many state-of-the-art techniques are now being developed to ensure security for CPSs throughout attack modeling and simulation building. For instance, a testbed for integrated evaluation of

large-scale CPS systems has been developed using a model-based integration approach and the IEEE High-Level Architecture (HLA) based distributed simulation software; it also involves a Hardware-in-the-loop simulator which helps secure the system against hardware attacks, which is important for the system to be complete in the hardware and software aspects [24]. F. Pasqualetti, et al. proposed an attack model, which generalizes the prototypical stealth, false data injection and replay attacks, to form a unified framework and advanced monitoring procedures for malfunction and attack detection. In another research work a novel method to model cyber-physical attacks in smart grid with hybrid attack graphs has been proposed. Other frameworks developed for Attack modeling contain addresses combined (dependent) vector attacks and synchronization/localization issues. The framework identifies the cyber-physical features specified by the security policies that need to be protected and can be used for proving formally the security of cyber-physical systems [25].

Regulatory Groups are also analysing the problem of cybersecurity for Cyber Physical Systems, like:

- NIST in US with a proposed Framework for Cyber Physical Systems [26],
- ENISA – European Union Agency – provided a set of Good Practices for IoT and Smart Infrastructures and a tool to support IoT operators and industries to conduct risk assessments [27].

4.4 Dependability

Dependability is a superordinate concept regrouping multiple system attributes, in various applications and domains different dependability attributes are highly prioritized and a key concept of modern embedded systems. Nevertheless, these different attributes, might lead to different targets or might lead to inconsistencies, if not appropriately covered. System dependability features have mutual impacts, similarities, and interdisciplinary values in common. System dependability attributes have a major impact on product development and product release as well as for company brand reputation.

4.4.1 Robustness

Robustness as a system property describes the degree to which a system is able to function correctly in the presence of disturbance, i.e. unforeseen or erroneous inputs. To study the effect of cascading failures and robustness in real social networks, H. Peng and his group [14] has constructed different CPS models consisting of interdependent physical-resources and computational-resource networks. They achieve promising results for various network builders to design a better network structure that can survive random network attacks. M. Rungger, et al. [15], introduce a notion of robustness termed input-output dynamical stability for cyber physical systems, which captures two intuitive aims of robustness: bounded disturbances have bounded effects and the consequences of a sporadic disturbance disappear over time.

4.4.2 Maintainability

In a cyber physical system, maintainability means that the system can be easily repaired after a malfunction, failure after a breach has occurred. We call a CPS maintainable if it is able to be repaired swiftly, with the minimum amount of expenses, and with no possibility of causing supplementary faults in the maintenance process. The best way to have a maintainable CPS is through rigorous monitoring and testing of the system's components and identifying the weak links in order to repair or replace them with higher-quality ones [11].

4.4.3 Safety

Available functional-safety standards tend to lead the development of safety-related systems towards fully dependent and closed systems in order to minimize the potential for fault propagation and limit complexity. However, the large number of intercommunicating nodes of CPSs limits the ordinary applicability of functional safety for the open environments of CPSs. CPSs require new approaches to real-time fault tolerance and reasoning about consequences of faults because the fault tolerance of CPSs cannot be solved solely as a software problem since these systems work on the tight coordination among hardware, software and physical elements.

Dependability has to meet safety as main requirement in systems engineering. To achieve dependability, CPSs have to counter possible faults by fault prevention, fault tolerance, fault removal and fault forecasting methods. Dependability and especially fault prevention methodologies have much in common with systems engineering approach, while fault forecasting has strong ties to safety analysis. Fault tolerance is used to avoid service failures when some part of the system fails, which is essential for guaranteeing the reliability of safety-related subsystems and limiting fault propagation. Fault removal is tightly coupled with the development process, and thus also with fault prevention.

4.4.3.1 Safety requirements in the Automotive context

Dependability case and safety case are systematic approaches that could be used to collect evidence for proving the dependability or safety of the system. In order to emphasize the importance of safety by system design in automotive, the definition of the Functional and TSC still remains an essential element as descripted in the ISO 26262. Whatever choice of the above safety methods on system design is made, that shall be justified according to the specification of technical safety requirements, as shown in the workflow at system level development shown in Figure 10.

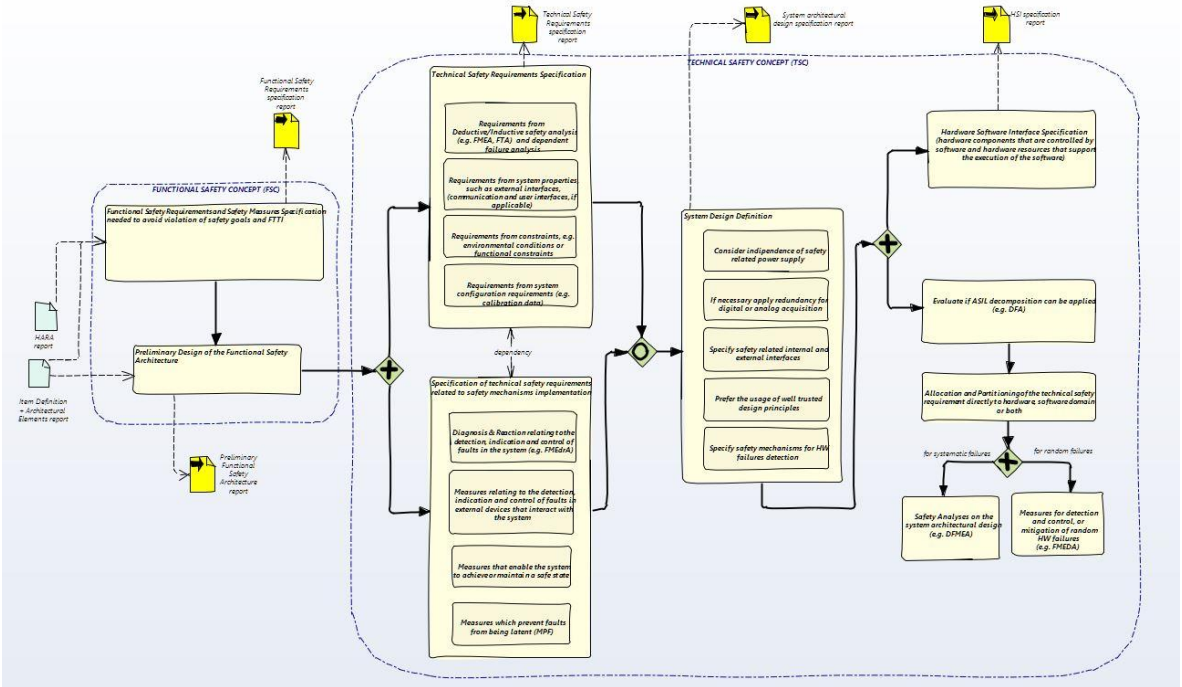

Figure 10: Specification of technical safety requirements Workflow

As a starting point for the risk determination according to the ISO 262626 a defined functionality including a first preliminary architecture shall be the basis. Therefore, the approach of the ISO PAS 21448 SOTIF can be used to support applying functional safety in accordance with the ISO 26262 – the creation of the item definition (see D5.1 - WP5). The item definition has to include a definition of the functions including their dependencies and interaction with the environment and other items/vehicles. Based on the item definition, a Hazard Analysis and Risk Assessment can then be carried out to find the root requirements or safety goals for the involved functions at vehicle level. Safety goals have to be clear and precise, do not contain technical details, but have to be implementable by technical means (e.g., avoid referring to non-measurable data). ISO 26262 requires that at least one safety goal is assigned to each hazard rated as ASIL A, B, C or D – the Automotive Safety Integrity Level is a measure of necessary risk reduction related to prevention or mitigation of the hazardous events (Figure 11). It is not necessary to define safety goals for hazards rated as "QM" or "no assignment", but hazards rated with QM shall be addressed by at least one requirement. One safety goal can address several hazards and a hazard can be addressed by more than one safety goal.

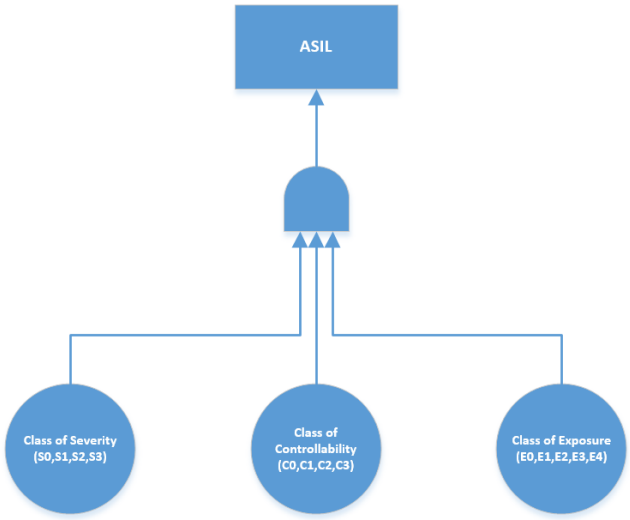

Figure 11: Safety Goal ASIL determination

Once the architectural system element of item has been defined, the next technical step will be to develop the safety concept (Figure 12). In addition to allow a correct and consistent deployment of safety goals within the item in the different abstraction levels of it, the safety concept will allow to find the proper safety measures that are needed to avoid violation of safety goals.

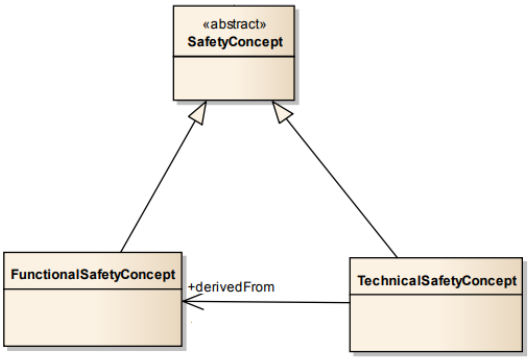

Figure 12: Safety Concept modelling with SysML

As mentioned above the Functional Safety Concept describes the safety measures that are needed to avoid violation of safety goals. It shall contain assumptions about necessary driver actions if needed to comply with at least one of the specified safety goals. It shall be available to start derivation of Technical Safety Requirements. These shall specify, if applicable, operations for:

a. fault avoidance;
b. fault detection and control of faults or the resulting malfunctioning behavior;
c. transitioning to a safe state, and if applicable, from a safe state;
d. fault tolerance;
e. the degradation of the functionality in the presence of a fault and its interaction with f) or g);
f. driver warnings needed to reduce the risk exposure time to an acceptable duration;
g. driver warnings needed to increase the controllability by the driver (e.g. engine malfunction indicator lamp, ABS fault warning lamp);
h. how timing requirements at the vehicle level are met, i.e. how the fault tolerant time interval shall be met by defining a fault handling time interval; and
i. avoidance or mitigation of a hazardous event due to improper arbitration of multiple control requests generated simultaneously by different functions.

If a safe state cannot be achieved within a defined fault tolerance time interval (FTTI), then a warning and degradation concept or strategy shall be specified.

The Technical Safety Concept (TSC) is derived by the Functional Safety Concept. It contains the refinement of the functional safety requirements and their allocation to components of item. The TSC refines the technical solution described in the Functional Safety Concept. The traceability shall be given from the safety goal, derived on vehicle level to the safety mechanisms specified in the TSC. The allocation of the safety mechanisms to HW component or SW component shall be described in the TSC.

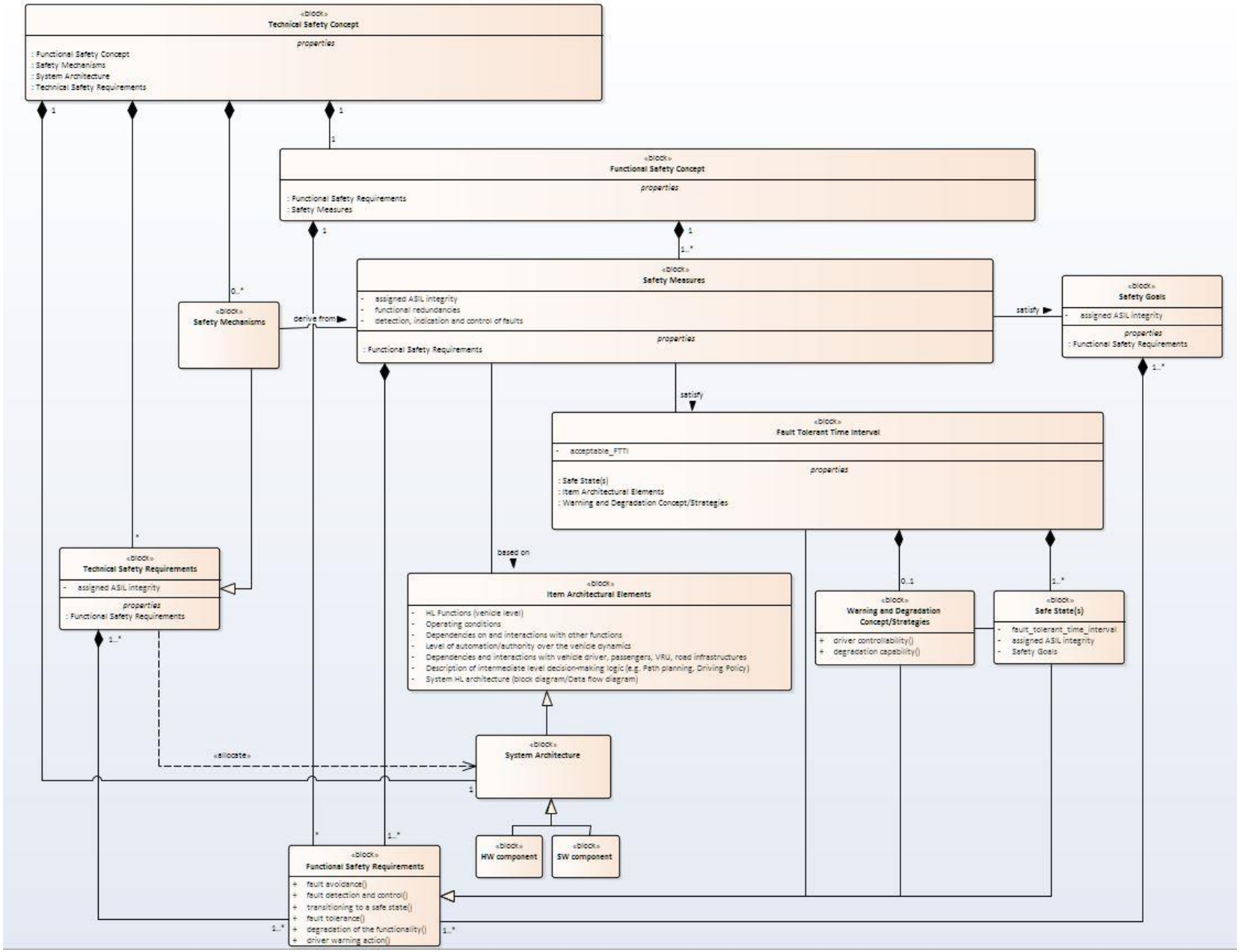

Figure 13: FSC and TSC modelling with SysML

There is also the case of potentially hazardous behaviour caused by the intended functionality or performance limitation of a system that is free from the faults addressed in the ISO26262 This is common in systems which rely on sensing the external or internal environment. Examples of such limitations include:

- The inability of the function to correctly comprehend the situation and operate safely; this also includes functions that use machine learning algorithms causing incorrect classification, incorrect measurements, incorrect tracking, misdetection, ghosts, incorrect target selection, incorrect kinematic estimation, etc;
- Insufficient robustness of the function with respect to sensor input variations or diverse environmental conditions as well as occluded field of view of environmental sensors.

The absence of unreasonable risk due to these potentially hazardous behaviours related to such limitations (including the human/machine interface) is defined as the safety of the intended functionality (SOTIF). Functional safety (addressed by the ISO26262) and SOTIF are distinct and complementary aspects of safety. SOTIF is to be applied to intended functionality where proper situational awareness is critical to safety, and where that situational awareness is derived from complex sensors and processing algorithms; especially emergency intervention systems and systems with levels of automation 1 to 5 on the OICA / SAE standard J3016 automation scales. Moreover, reasonably foreseeable misuse, which could lead directly to potentially hazardous system behaviour, is also considered as a possible condition that could directly trigger a SOTIF–related hazardous event.

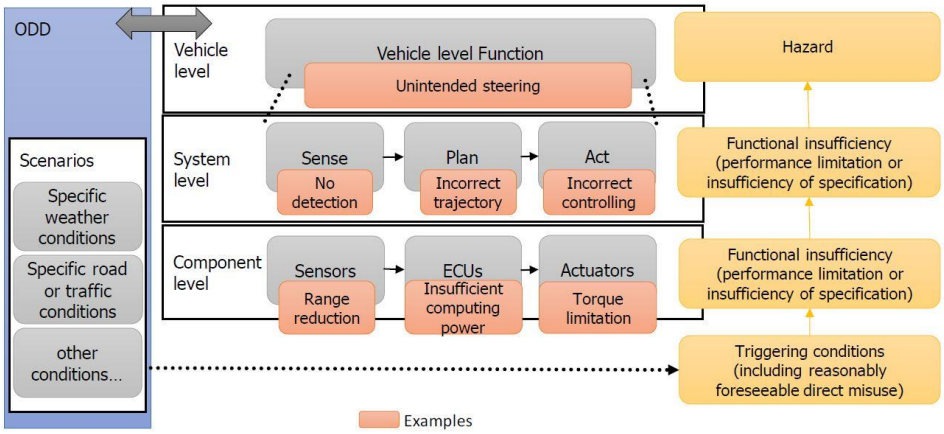

Figure 14: Links among hazard, triggering conditions and the system performance limitations

The alignment of the SOTIF and ISO26262 is important to implement possible modifications to the system design at a sufficiently early stage. The beginning of the SOTIF development process keeps aligned with the Item definition and the Hazard Analysis and Risk Assessment of ISO26262 process. 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 Functional Safety Concept and TSC of ISO26262 processes. Verification and Validation of the SOTIF are always aligned with the corresponding ISO26262 activities on the right side of the V-model. Definition of the SOTIF V&V strategy is already compiled from information produced in the early stages of the SOTIF development. SOTIF Release and Functional Safety Assessment conclude the development process.

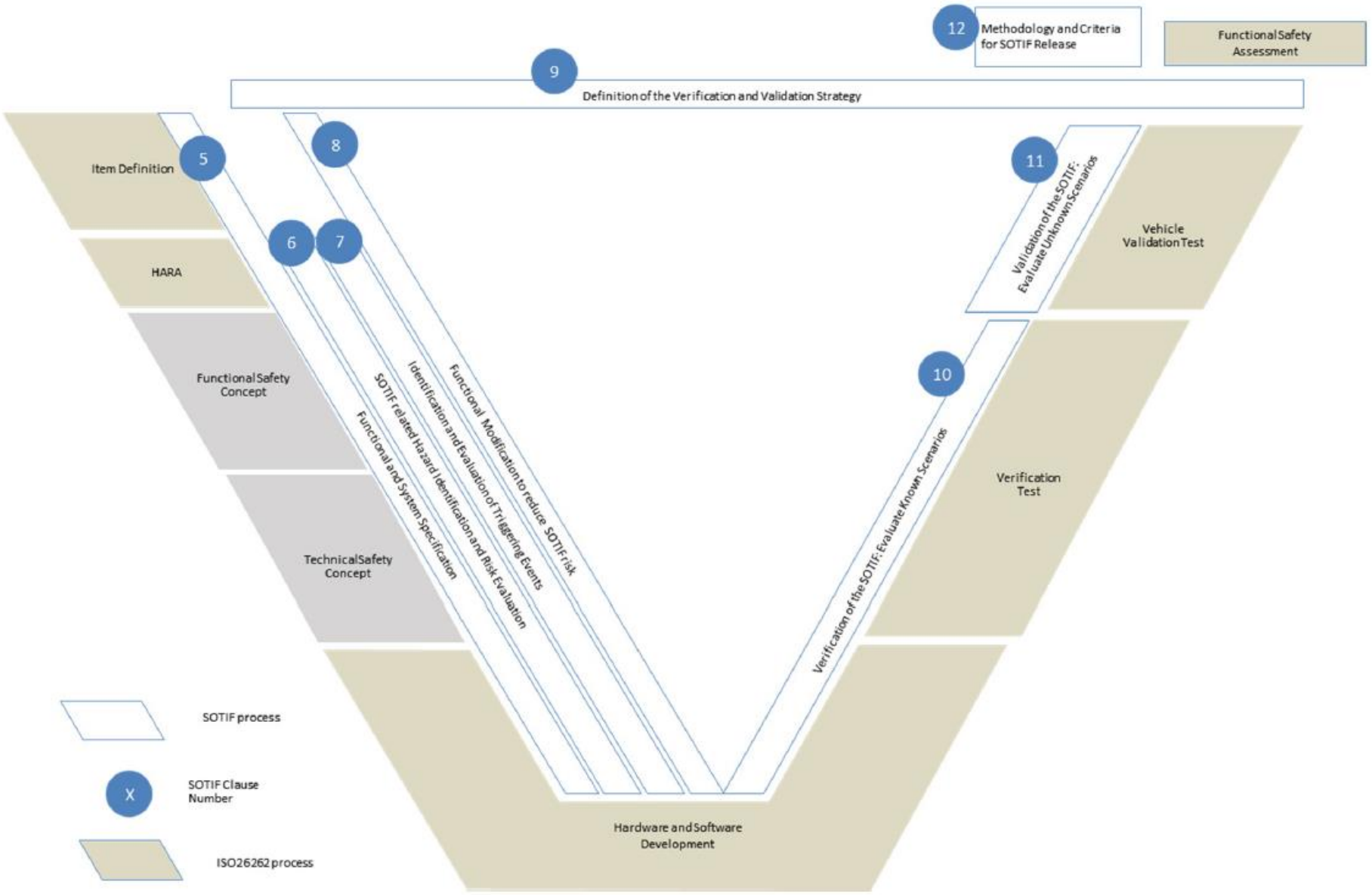

Figure 15: Interactions of Product Development activities between SOTIF and ISO26262 processes

4.4.3.2 Safety requirements in the Avionic domain

This section identifies the overall safety-related requirements from the avionics domain. We'll first introduce the terms, and then focus on the reliability and timing aspects of avionics safety-critical systems.

To present the basics of timing, safety and reliability concepts in avionics, we use the following definitions from the field of fault-tolerant avionic software, as presented in [8], [28].

- An **element** is used in a general way, according to IEC61508 [29], to describe either hardware, software, or hardware/software combination or an entire system or subsystem.
- The **total state** of a given element is the set of the following states: computation, communication, stored information, interconnection, and physical condition. The part of the element that is perceivable from the outside of the element is the **external state**; the remaining part is the **internal state**
- A **service** delivered by an element is its observable behavior. The delivered service can therefore be defined as a sequence of the element's external states.
- A **failure** is a deviation between the delivered service of an element and the correct expected service of this element. It reflects the inability of an element to perform correctly its intended function within the specified performance requirements. A service failure means that some external states deviate from the correct service states.
- An **error** happens when an internal state of an element deviates from its expected correct state. An error can also cause a failure if the external state of the element is affected. However, most errors are captured before impacting the external state, not causing a failure.
- A **fault** is defined as the cause of an error. It could either be a defect in the hardware components of the element, or an active bug in the element's software.

In the avionic domain, **Reliability**, is defined as the ability of an element to provide the correct service over a given period of time, or in short as the **continuity of service**. In error-prone systems it is the role of fault-tolerance mechanisms to prevent errors from producing failures and to assure reliability. Errors might be caused by different types of faults, amongst others by **random hardware faults**. Modeling random hardware faults with statistical **error models** facilitates **reliability analysis**, i.e., the determination of an **element**'s reliability analytically.

While reliability has been originally applied as a measure to characterize behavior of an element in the value domain, **Timing** refers to the element's behavior in the temporal domain. Timing can include various properties, for example minimum, maximum or average latencies, computation and communication jitter, process activation patterns, deadlines and so on. In the temporal domain, an **error** occurs when detecting that the above-mentioned timing constraint will not be respected (such as a future deadline miss). Corrective actions usually include re-scheduling actions like priority change of the task with endangered deadline. If such an error causes the late delivery of an element's service, it also causes a **failure**.

The role of **Safety** is slightly different. In very general terms safety is defined as the **freedom of unacceptable risk** [29]. More concrete approaches define safety as absence of catastrophic consequences on the user and the environment. Anyway, each of these definitions refers to some kind of non-tolerable events, generally called **hazards**. Hazards have to be identified at design time and avoided at runtime by corresponding safety measures. In safety-critical systems, if **failures** occur, the system safety functions must become active to make sure that

neither a hazard occurs (e.g., by transition to a dedicated state which provides degraded, but safe service) nor failures are propagated in an uncontrolled manner (e.g., using isolation mechanisms).

Safety process in Avionics

Both Reliability and Timing fault tolerance mechanisms deal with possible internal state deviation such as transient hardware fault, permanent hardware fault or missing an internal deadline. After detecting fault-induced errors, they are using correction mechanisms (e.g., CRC based fault correction, task redeployment, …) to fall back into correct service mode, trying not to affect the external state. Figure 16 presented below details at which level faults, errors and failures are dealt with in the safety process.

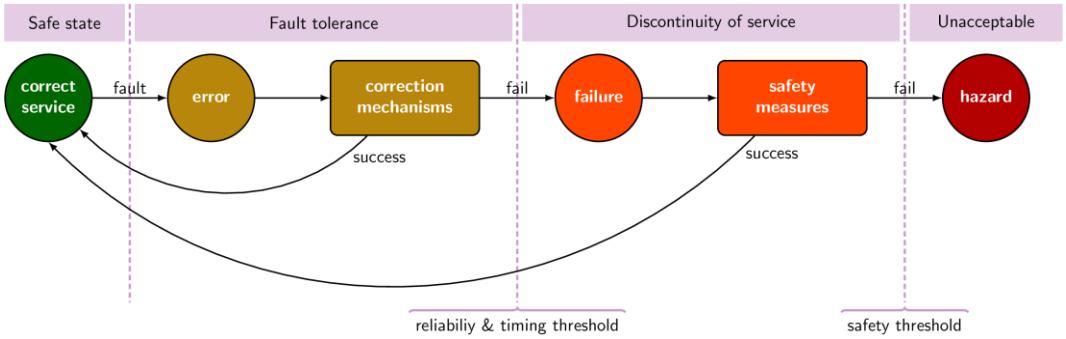

Figure 16: Avionics safety process

When a fault tolerance mechanism fails and the external state is impacted, it causes a failure implying a discontinuity of service. Safety measures then allow this failure not to propagate to the whole system and also that a hazard does not occur. Safety correction measures (reinitialization, reset, switch to degraded mode) are then taken so that the system could fall back into a safe state. The failing of safety correction measures will lead to some hazard and is usually not acceptable.

Design Assurance Levels

In the DO-178-B standard, safety levels are represented with **Design Assurance Level** (DAL) and are defined in terms of the impact and maximum probability of a failure on the flight. The safety levels range from DAL-A where the effect of a failure will lead to catastrophic consequences such as plane crash or loss of life, to DAL-E where the effect of a failure will have no impact on the plane such as impacting the on-flight video of the passengers. Safety levels are presented in Table 4 including the maximum allowed probability that such an event occurs.

Table 4: Design Assurance Level in Avionics

Level	Severity	Consequence	Probability (per hour of flight)
A	Catatrophic	Failure may cause a crash. Error or loss of critical function required to safely fly and land aircraft.	$<10^{-9}$
B	Hazardous	Failure has a large negative impact on safety or performance, or reduces the ability of the crew to operate the aircraft due to physical distress or a higher workload, or causes serious or fatal injuries among the passengers.	$10^{-7} \rightarrow 10^{-9}$

C	Major	Failure is significant but has a lesser impact than a Hazardous failure (for example, leads to passenger discomfort rather than injuries) or significantly increases crew workload.	$10^{-5} \rightarrow 10^{-7}$
D	Minor	Failure is noticeable, but has a lesser impact than a Major failure (for example, causing passenger inconvenience or a routine flight plan change)	$10^{-3} \rightarrow 10^{-5}$
E	No Effect	Failure has no impact on safety, aircraft operation, or crew workload.	$>10^{-3}$

Figure 17 provides some example of avionics systems with their respective design assurance levels. For the purpose of the TEACHING avionic use-case, we will consider the FMS application to be DAL-B, while the cyber black-box application to be DAL C/D.

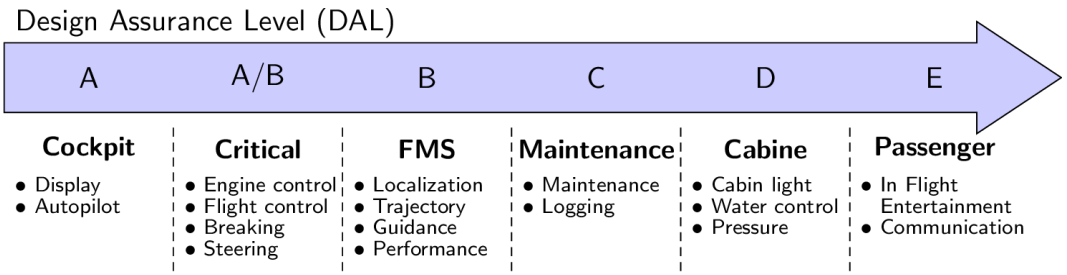

Figure 17: Design Assurance Level of avionic systems

Spatial and Temporal Partitioning

To allow several software application to run on the same system, the avionic industry heavily relies on **strict partitioning** following the ARINC 653 principles [30], as depicted in Figure 18. The ARINC 653 (Avionics Application Standard Software Interface) software specification defines space and time partitioning in Safety-critical avionics real-time operating systems, allowing the industry to host multiple applications of different software levels in the same hardware in the context of Integrated Modular Avionics architectures [31].

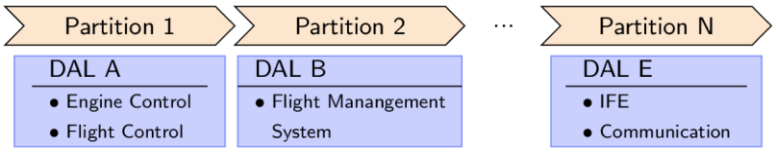

Figure 18: ARINC partitioning

Each application fits in a partition and has its own memory space, as well as a dedicated pre-allocated scheduling time slot registered in the real time operating system, which also manages inter-process / inter-partition communication and error handling.

Spatial partitioning ensures that it is not possible for an application to access the memory space (both code and data) of another application running in a different partition. Robust partitioning includes the protection of each partition memory space. This is usually provided by hardware memory protection mechanisms, i.e. MMU. It also requires a functional protection concerning the management of privilege levels, and restrictions to the execution of privileged instructions.

Temporal partitioning ensures that the activities of an application in one partition, including missing a deadline, do not affect the timing of the activities of another application in another partition. Ensuring temporal partitioning is a protection against timing anomalies. Temporal partitioning is actually a part of a more general property – the time composability. The latter property also requires the temporal independence between different segments of the same application, so that analysis can concentrate on the longest (worst-case) execution path and unpredictable hardware behavior, such as domino effects, can be avoided.

4.4.3.3 Time-critical requirements in the Avionic domain

In avionics, safety-critical applications are characterized by stringent real-time constraints, making **time predictability** a major concern with regards to the regulation standards [32], [33]. Furthermore, avionics real-time computing systems are characterized by the fact that the correctness of an operation or a task is not only defined by its functional correctness, but also by time-window during which those operations or tasks have to be executed.

However, the recent shift to multi-core processor, and now the shift to heterogeneous architectures with AI accelerators is introducing new sources of time variations. As a consequence, the industry is facing a trade-off between performance and predictability [34], [35].

Worst-Case Execution Time (WCET)

A common practice to guarantee the time predictability and its associated deadlines of a time-critical application with single-core architecture is to determine the application Worst Case Execution Time (WCET). This WCET computation usually relies on analysis tools based on static program analysis tools [36], detailed hardware model, as well as measurement techniques through execution or simulation [37]. However, these analysis techniques and tools are not currently able to provide an exact computation of the WCET, only delivering an estimated upper bound, introducing some safety margins as depicted in Figure 19.

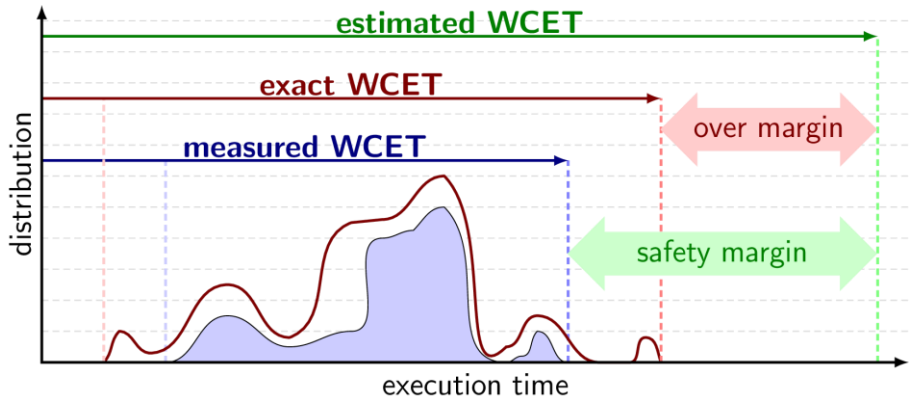

Figure 19: Worst Case Execution Time & Safety Margins

Despite all the improvements in the WCET estimation domain over the last decades, the overestimation remained mostly constant as the predictability of the architecture decreased [36]. This makes the use of WCET analysis tools difficult for real industrial programs running on multi-core Commercial off-the-shelf (COTS) architectures [34], [35].

Multi-core architectures and the issue of Timing Interference

Several studies [38], [39] have shown that the order of magnitude on the variation of the maximum observed execution time while using the 8 cores of a multi-core architecture was

larger than the expected gain from using a multi-core (with up to a 20x on the worst case for 8 cores).

The source of this slowdown has been identified: On a multi-core processor, different pieces of software will be executed on different cores at the same time. Such different software will compete electronically to use the shared hardware resources of the processor architecture, causing concurrent accesses to the same hardware.

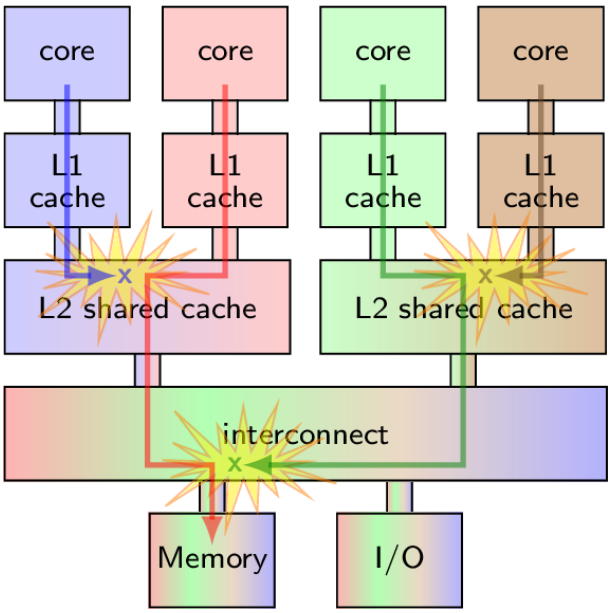

Figure 20: Timing interference in multi-core systems

On the hardware resources side, concurrent accesses are arbitrated, introducing inter-task or inter-application jitter defined as **timing interference** [38] as illustrated by Figure 20. These interference are breaking the timing isolation principles required by the standards [29], [32], [33] of avionics safety-critical software. A consequence for the industry is to decide between a trade-off in terms of performance versus time predictability [34], [35].

Mitigation techniques to deal with timing interference exist in the literature [40] as **deterministic platform software** (DPS), and each technique is proposing a different trade-off in terms of performance, time predictability, and maturity level with regards to the industrial practices of the avionic domain.

4.4.3.4 Reliability

As defined by Avizienis et al. [8], reliability describes the continuity of correct service. Reliability engineering focuses on the ability of a system to function without failure. Reliability is closely related to availability and often confused with availability or safety.

While reliability engineering deals with the prediction, prevention and management of uncertainty and risks of failure of the system, safety focuses on minimizing the potential for fault propagation and limit harm to people and environment. Reliability engineering is concerned with overall minimisation of failures that could lead to losses. Therefore, reliability engineering is more closely related to Quality Engineering. Whereas safety engineering focuses on minimising failures that could lead to loss of life, injury, or damage to the environment.

Reliability functions of components are defined as depicted in Figure 21and provide a probability indication that the component is working until time t.

Reliability R(t) of a component:

the **Probability** that it will survive at least until time t

$$R(t) = \frac{N_{ok}(t)}{N} = \frac{N_{ok}(t)}{N_{ok}(t) + N_f(t)} \qquad Q(t) = \frac{N_f(t)}{N} = 1 - R(t)$$

$$R(t) = 1 - Q(t) = 1 - \frac{N_f(t)}{N}$$

Looking at Interval (t_0,t):

N … # of components

N_{ok}(t) … # of components that operate correctly at time t

N_f(t) … # of components that have failed at time t

Figure 21: Reliability function of a component

In the context of industrial IoT systems, reliability can also be defined as the ability of the system to correctly deliver sensor data and actuation commands throughout a network, or as the ability to meet timing constraints in a changing environment. The ever-increasing demand for the highly dynamic reconfiguration of such systems pushes the demand for reliable connectivity and adaptation solutions. Therefore, in this context reliable systems are considered to be (runtime) adapting systems and systems that can maintain their functionality also in the presence of changing environment.

4.5 Acceptability

TEACHING research aims at filling the gap between AI and humans by means of a novel human-centred approached applied to the design and development of dependable AI. Human-Machine interaction is a rapidly growing research topic. Thanks to the AI and Industry 4.0 hype, a lot of researchers started to realize that technology alone isn't enough. In order to innovate processes, services and technology we need to build a usable technology that goes beyond the simple improvement of performance and increasing of available features. In order to do this, it is necessary to consider usability as a main requirement to be taken into consideration since the preliminary design phases.

The application of Human-Machine Interaction Research to the design of new products and service is mainly based on two complementary methods: Human-Centered Design and Design thinking.

Design Thinking is a problem-solving process that starts from the understanding and scoping of a clear problem and then focuses on how to solve it (Figure 22).

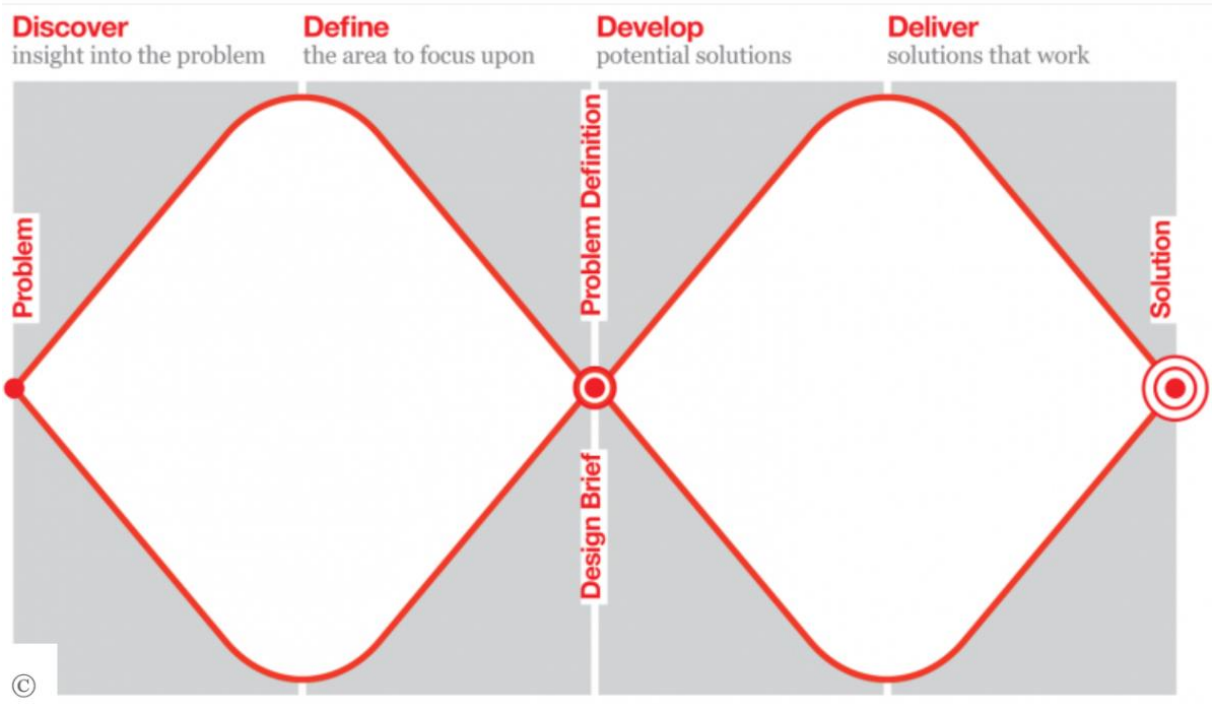

Figure 22: the Design Thinking process. (source: Design council)

Discovery is the beginning of the process, which looks to understand the 'problem': The user and their environments, behaviours, tools they use and decision-making processes. It looks at the landscape somewhat broadly often through methods such as surveys, user diaries, observations or immersion.

The idea is to really understand the user to build empathy.

The "Define" phase is an opportunity to refine and narrow down ideas and to look at the main challenges the product/service may face.

The final two stages of the Design Thinking process are "develop" — the ideation stage where design concepts are tested out — and "Deliver" — where the product is finalised and launched.

The double diamond (Figure 22) isn't the only representation of Design Thinking out there. Another alternative way of understanding the process is through Stanford Design School's Design Thinking Bootleg (Figure 23).

Figure 23: The Stanford Design School's Design Thinking Bootleg. Source: Medium[7]

Human-centered design is an approach to problem-solving that has a focus on the people that you're designing a product for. It starts by establishing who your user is and what their problem is and then ends by finding a solution that is tailored to them.

The objective of Human-Centred Design (something referred to as "HCD") is to help designers in producing an abundance of ideas. These preliminary ideas will feed the process moving into the building step and then sharing a prototype with the ideal customer and, eventually, putting the product or service to market.

Human-centered design consists of three phases, shown in Figure 24.

[7] https://medium.com/@petrila3/bootcamp-bootleg-f7132a181db1

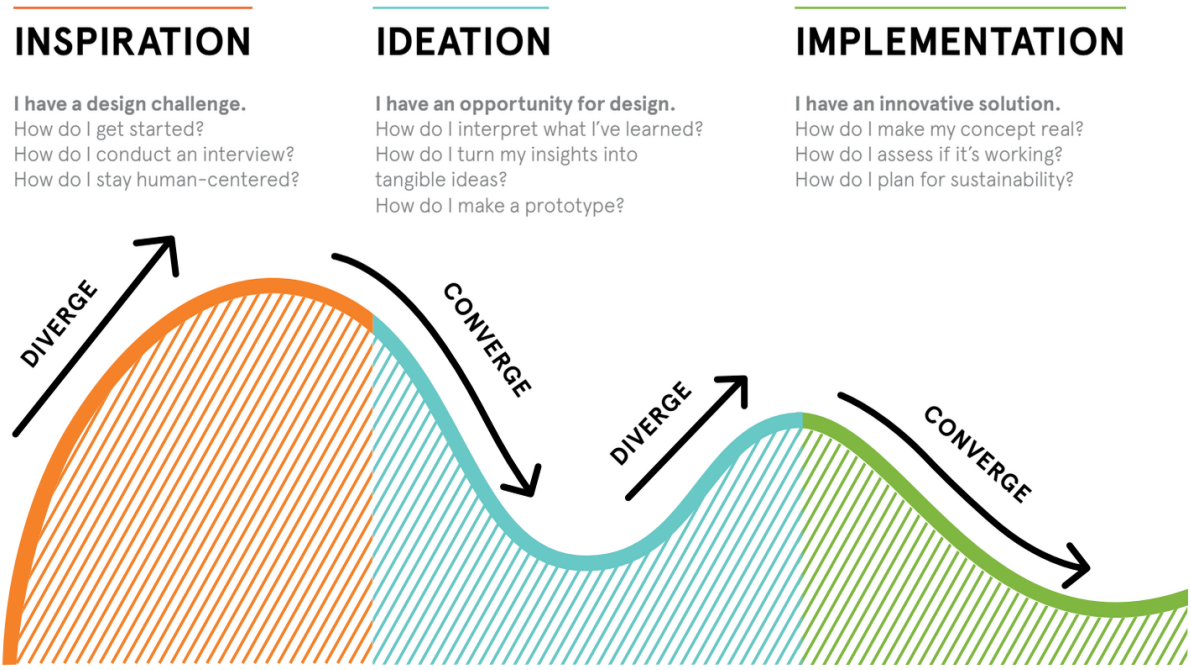

Figure 24: The Human Centered Design flow. (source: Medium[8])

During "inspiration", designers focus on learning from the people you're design for. To build empathy with them and better understand their needs.

In the "Ideation" phase, it is important to bring order to what has been learnt. It begins by mapping out opportunities for design and prototype solutions to the problems uncovered in the previous phase.

Finally comes "Implementation". This is when the solution is built. Trusting the human-centered nature of the design process the solution has an higher probability to be well accepted by the final users thus by the market.

IDEO[9] suggest that this mindset can be split into these actionable steps:

- Observation: Learning about the end-user, through research
- Ideation: Brainstorming a lot of ideas
- Rapid Prototyping: Quickly building a low-fidelity prototype
- User Feedback: Getting input from your end-user
- Iteration: Iterating and fine-tuning the product and its design, whilst continuing to get user input.
- Implementation: Test the idea out in the real-world

Design Thinking and Human Centered Design methods are not mutually exclusive. The first is a project execution template that help in keeping the design focus centered on the user need. Human Centered Design is a design and thinking methods that can be applied iteratively to all the passes of Design Thinking in order to develop solution by keeping humans at the centered of each phase of the product design and development phases.

[8] https://medium.com/snapout/design-thinking-vs-human-centred-design-whats-the-difference-9ef855f55223#:~:text=Design%20thinking%20looks%20at%20the,a%20particular%20product%20or%20service.
[9] http://ideo.com/

In TEACHING, the Design Thinking method will be suggested to all the partner involved in the design and development of AI components as framework for the organization of the activities or at least as reference for the validation of the design process. Moreover, the Human Centered Design "forma-mentis" will be promoted among the consortium teams in order to guarantee a design process highly focused on users' needs and requirements.

4.6 Energy efficiency

This analysis was conducted placing IFAG's demonstrator at the center. All aspects of energy efficiency regarding this demonstrator can be optimized by analysing the following questions and adapt the structure due to the results of these key questions:

- Are tasks executed which are not needed? Due to continuous improvements and extended functionality of products a lot of times obsolete modules are dragged along though not providing any surplus to the functioning of the CPS.
- Can tasks be scheduled with lower frequency? To reduce the overall load of the hardware it is essential to determine the lowest frequency a task or module has to run or be updated by providing the intended functionality. A higher task frequency than the minimum will result in higher energy consumption.
- Can the task be executed sufficiently successful with fewer resources? During the optimizations of applications, it is not only important to guarantee a high accuracy depending on your task, but also take into account if the needed outcome could also be achieved with less resources. For image processing task for example that would correlate with the lowest resolution the application could be run with while achieving their goal. Or with on edge neural network setups a smaller number of nodes could lead to a less memory consuming application
- Are overlaps in between tasks resulting in multiple processing? Especially for large CPSoS it is important to have a well-documented structure to prevent that functions partly overlap and thus not only lead to a larger program but also result in the same calculations being done multiple times, at different parts of the system, instead of storing and reusing the result, thus saving the energy of unnecessary repetitions of code.
- Is there dedicated hardware optimized for the needed task? A lot of times it is not necessary to reinvent the wheel. Coming back to on edge neural network CPS for example, there are specialized chips that are highly optimized also with a focus on energy consumption, to run predesigned networks.
- Does a less energy consuming platform satisfy the needs of the tasks? As the development of CPSoS are usually a iterative process it is also important to doublecheck if the overall system would fit on a hardware platform that might not be as powerful but sufficient for the task. Though also considering future developments that might ask for a more powerful system.
- Are things modular and scalable to adapt to future requirements. Having a system that is highly modular makes sure that only the amount of actually needed hardware could be used. To give a more concrete example you could think of a sensor system detecting the stress level a person in a vehicle. Due to the different number of passengers the number of needed sensors would vary highly between a bus or a five-seater personal car. Thus, a modular system would give the chance to adjust exactly to each vehicle. Though, during the design of any scalable system it is important to consider the boundaries of scalability, due to for example, network interconnection limits. But usually, modularity speaks in favor of energy efficiency.

Apart from the earlier mentioned key questions that mainly focus on the software design point of view, there is plenty of research among which the work of [41] gives an insight of dealing with the topic of efficiency in embedded systems with a focus on hardware solutions.

Some of those topics are especially interesting for the TEACHING approach and therefor will be explained in detail.

Power consumption can be split in two parts. The static part that originates from leakage and the dynamic part that is produced due to the switching thus the calculation. There are different optimization methods for both parts.

The reasons why focusing on energy efficiency is important are the following:

- Limits due to battery capacity,
- Reduce the heat generated by the system,
- Overall cost reduction,
- Ensure longevity,
- Avoid overprovisioning of resources,
- Meet performance requirements,
- Miniaturization leads to smaller fractions on the chip that can run at full speed due to the power budged,
- Global trends of the amount of edge devices,
- Moral point of green computation.

The techniques to save energy can be split in the following approaches:

Dynamic voltage and frequency scaling (DVFS).

If the system is running on a lower frequency it can be operated at a lower voltage which leads to less energy consumption. This way dynamic loads can be dealt with. Downside of this method is the decrease of performance and hence its increase in execution time, which might lead to missed deadlines. This method also requires hardware, which is supporting variable clock frequencies and voltages. The benefit of these methods is small due to an increase of leakage energy and multi-core processors structures. A vast amount of different algorithms how DVFS can be applied are given in [41].

Power mode management

Save energy by setting the system in different modes that are implemented by design. In an idle state the system might select a low-power mode, waiting until it is actually needed to perform tasks in a normal mode. It is easier for the programmer to use than DVFS as no scheduling of tasks and surveillance of deadlines is required. Mittal in [41] mainly evaluates different algorithms to calculate break-even time calculations, at which a switching of power modes results in a beneficial energy saving.

Microarchitectural techniques

The microarchitectural techniques focus on different possibilities to reduce energy regarding memories or caches. All sorts of algorithms that compress data such that less physical memory is needed might result in energy savings if the energy reduction due to savings of memory exceeds the energy needed for the compression. Also, special buffer structures might lead to energy savings in read-/write-combining prefetching schemes. In case of dynamical usage of cache due to different programs or program phases the unused cache could be switched off, which leads to energy savings. Alternative memory types can also be less energy consumption but consumes more time.

Specialized hardware (GPUs, FPGAs, ASICs)

General purpose processors are by nature designed to be able to execute all sorts of calculations. Unconventional cores are often optimized for a single task only and due to this specialization require less energy at a faster throughput. By combining different hardware types, the overall system could consume less energy. The major downside of the hardware accelerators is the need for parallelization and high peak power consumption. Also, the high specialization makes this approach beneficial for certain areas of application only.

5 Baseline technologies and tools

Following the establishment of the TEACHING R&D work for meeting the NFRs, we also present a survey on the tools and technologies that are readily available and that could form the baseline of the TEACHING Platform. We focused on the hardware platform upon which TEACHING could build its platform.

Recall the list of high-level functional requirements from Section 3.2:

Table 5: List of functional requirements

ID	Title
FR1.1	Integrate with the onboard ADAS and interact with it
FR1.2	Communicate with the wearable sensors
FR1.3	Communicate with the web
FR1.4	Execution of software directly or in the form of container images
FR1.5	Execute machine/deep learning algorithms for training
FR1.6	Execute machine/deep learning algorithms for inference
FR1.7	Offload tasks at the edge nodes and generally enable an interplay between local and remote resources
FR2.1	Integrate with the onboard FMS and retrieve data from its monitoring probes
FR2.2	The system must allow the execution of software directly or in the form of container images
FR2.3	Execute machine/deep learning algorithms for training
FR2.4	Execute machine/deep learning algorithms for inference
FR2.5	It must also be able to communicate the results of its processing to the human pilot

The implication of those requirements (Table 5) is that the TEACHING CPSoS is an **embedded computing platform** that may connect to other suchlike devices or be able to execute concurrently the TEACHING applications and dependable systems like the ADAS or the FMS (FR1.1, 2.1). The computing platform must be able to communicate with **sensors** and the web, i.e. implement standard communication protocols (FR1.2, 1.3, 2.1) and it must be able to support containerization (FR1.4, 2.2). The platform must also be able to support ML/DL training, perhaps through the appropriate **software tools** and a GPU (FR1.5, 2.3) and inference (FR1.6, 2.4). Furthermore, an execution management environment must exist that orchestrate underlying resources (FR1.7). Finally, some libraries for human-machine interaction must be provided (FR2.5).

Following this analysis, we present an overview of the COTS **hardware platforms** (Section 5.1), **sensors** (Section 5.2) and **software tools** (Section 5.3) that fit the above mentioned, highlighted criteria. For each of those we also present the tradeoffs it achieves in meeting the NFRs.

5.1 Hardware platforms

In this section, some of these platforms available on the market for the needs of the TEACHING CPSoS applications are listed and described.

5.1.1 Nvidia Jeston Nano

Nvidia is a big player in the GPU market and recently moved to the AI field where high parallelism brings increased performances. The Jeston Nano is a small platform for prototyping IoT applications that include AI. It is a Quad-core ARM Cortex-A57 MPCore processor but the strong point is in its NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores.

It is not built for reliable and safe applications, but more as a developing kit for research and prototypes.

5.1.2 Nvidia Drive AGX Pegasus

This is the top-level product from Nvidia, specifically targeting the L4 and L5 autonomous vehicle functions. It has two Xavier SoCs and two discrete NVIDIA Turing™ GPU. The platform is designed following safety standards and has some redundancy available since it has two SoCs and two GPUs.

The strong point of this platform is for sure the computing power available. The drawback is a very large power consumption that can reach 300W and a large cost [42].

5.1.3 Raspberry PI 4

The Raspberry Pi is a popular Single Board PC popular in the hobbyist community. It has a very large user base and has a large amount of available software. The downside is that targets the consumer market, so it does not have safety features and has no powerful AI accelerator.

5.1.4 Zynq Ultrascale+

While not a proper platform, this SoC has many benefits that may be useful for the project. It is an SoC with 4 ARM cores, 2 Cortex R5 cores, and programmable logic, which can deploy an AI accelerator with high performance and low power consumption.

5.1.5 I&M SDF Platform

In past projects, I&M developed a platform for Sensor Fusion and Autonomous Driving applications. Being developed with safety features in mind, it relies on two different processors: one is the ASIL D Aurix processor and the other one is high-performance SoC. The platform has been developed as a System On Module (SoM) hosting the SoC and a carrier board hosting the Aurix. I&M developed a SoM featuring a i.MX8 Quad Max, that is a SoC from NXP targeting infotainment applications.

The SoM and the carrier are designed so that they both follow the SMARC standard from SGET, in this way it is possible to swap the SoM with the i.MX8 with another one. For instance, a SoM with an Ultrascale Zynq+ MPSoC can be used, delivering both Cortex-A cores and programmable hardware that can be used for NN acceleration.

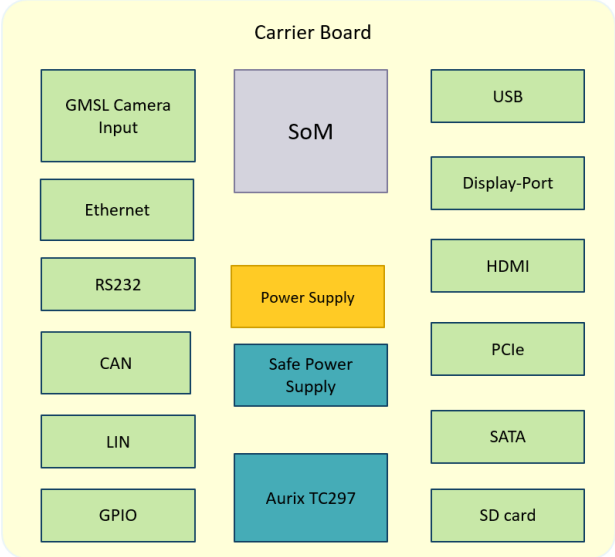

Figure 25: Carrier Board components

The carrier board (Figure 25) has several interface, from CAN to LIN and Automotive Ethernet that rely on two Ethernet switches that bridge the two processors while expanding the number of external ports.

The Aurix processor, with lockstep cores, together with its power supply TLF35584 can guarantee the ASIL D safety level. The TLF35584 acts as a watchdog for the Aurix. Moreover, the Aurix can check the status and healthiness of the system, including the other processor.

5.1.6 ARMv8-based NXP iMX8 Quad Max

While not directly embedding a custom AI accelerator, the heterogeneity from the ATM bigLITTLE technology provides the SoC with different compromise in terms of performance over power consumption ratios.

For instance, the iMX8 Quad Max from NXP embeds two ARM Cortex M4 real-time microcontrollers, four ARM Cortex-A53 low power cores, and two Cortex-A72 high-performance cores. This is furthermore complemented by a GPGPU making this architecture a viable choice for dealing with both the predictability requirements of dependable systems, and the high performance computing requirements of AI algorithms.

5.2 Sensors

5.2.1 Sensors for Software & Hardware Monitoring

Most recent architectures are providing some special hardware allowing us to assess the hardware behavior: the hardware **Performance Monitoring Counters (PMC)**. Such counters are usually counting the occurrence of architectural events (such as number of caches misses, accesses to the bus, and so on…) and therefore provide performance information [43] on the applications, the operating system, and the underlying hardware behavior.

Performance Monitoring Counters were initially designed for performance debugging purpose, providing the programmer with better tuning the applications to maximize their performance through various optimizations.

But such information is also critical in a safety-critical context: It allows to characterize workloads, allowing us to quantify hardware resource utilization at application level and providing some clues to better understand runtime variability [44]. In a multi-core or many-core context, hardware monitors are an opportunity to observe contention phenomena at the level of the shared hardware resources.

Performance Monitoring Counters are implemented as special purpose registers available from the microarchitecture instruction-set are usually confined to count on-die related events. Some architectures however also provide some monitoring facilities at platform / SoC level, allowing us to also monitor DRAM memory and peripheral accesses.

One of the project consortium members, namely TRT, developed METrICS [45], a measurement environment for time-critical systems that is developed on top of SYSGO PikeOS to provide an accurate measurement of timing and shared hardware resource accesses performed by the software. METrICS relies on PMC to observe the behavior of the software with regards to shared hardware resource, while minimizing the timing intrusiveness of the measurement environment.

METrICS was used to perform statistical analysis of the traces of collected hardware events to helps the expert to characterize the hardware platform with regards to safety or security constraints. In the TEACHING project, beyond porting METrICS to a new hardware and a new operating system, we plan to direct these traces toward AI Deep Neural Networks (DNN) to learn the expected behavior of the software with regards to the hardware to later detect safety or security issues as deviation from the expected behavior.

5.2.2 Sensors for Human Monitoring

A central feature of TEACHING is the integration of the human-in-the-loop of the autonomous application, which requires monitoring their physiological, emotive, and cognitive (PEC) state. This has been one of the key motivational aspect in the selection of sensor devices, whose integration will be explored as part of the project activities. In particular, we have 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.

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 the following, we provide the list of devices selected based on the criteria described above and grouped based on the type of sensed information.

5.2.2.1 Inertial Data

Device - Shimmer3 Inertial Measurement Unit (IMU)

Link - http://www.shimmersensing.com/products/shimmer3-imu-sensor

Description - Wearable wireless sensor device with integrated 9 degrees of freedom inertial sensing, encompassing via accelerometric, gyroscopic, magnetic and pressure sensors, each with selectable range. The device also includes an embedded motion processor for 3-dimensional orientation estimation. All signals can be measured simultaneously and in real-time. Collected raw-data can be logged on an onboard SD-card or streamed through Bluetooth connectivity.

5.2.2.2 Cardiac data

Device - Shimmer3 Electrocardiography Sensor (ECG)

Link - http://www.shimmersensing.com/products/shimmer3-ecg-sensor

Description – Wearable wireless sensor device for the measurement of ECG signals supporting bipolar limb leads and V1-V6 configurations of the electrical probes on the subject. The device also integrates inertial motion sensing capabilities and provides signal validation functionalities, including test signal generation, respiration demodulation and lead-off detection. The supporting software includes algorithms for heart-rate estimation from ECG signals that can be applied in live streaming data or to logged information. Collected raw-data can be logged on an onboard SD-card or streamed through Bluetooth connectivity.

5.2.2.3 Myographic data

Device - Shimmer3 Electromyogram Sensor (EMG)

Link - http://www.shimmersensing.com/products/shimmer3-emg-sensor

Description – Sensor for the measurement of the EMG signals integrated into the same wearable device hosting the Shimmer3 ECG unit. It performs a non-invasive EMG that records the electrical activity associated with contractions of the whole muscle and assesses nerve conduction. The device allows recording of two channels of EMG data that can be measured simultaneously with inertial data. The same device hosts the ECG sensor, but its recording is alternative to EMG. Collected raw-data can be logged on an onboard SD-card or streamed through Bluetooth connectivity.

5.2.2.4 Electrodermal activity

Device - Shimmer3 GSR+ (Galvanic Skin Response)

Link - http://www.shimmersensing.com/products/shimmer3-wireless-gsr-sensor

Description – Wearable wireless sensor device providing connections and preamplification for one-channel Galvanic Skin Response data acquisition (Electrodermal Resistance Measurement - Electrodermal Activity). The sensor allows monitoring the electrical characteristics or conductance of skin between two reusable electrodes attached to two fingers of one hand. It also integrates a component for capturing photoplethysmogram signals (either by a finger or ear-lobe optical probe), which can be used to estimate heart rate (HR), and an inertial sensor. All signals can be measured simultaneously and in real-time. Collected raw-data can be logged on an onboard SD-card or streamed through Bluetooth connectivity.

5.2.2.5 Brain activity

Device - EMOTIV Insight 5 Channel Mobile Brainwear

Link - www.emotiv.com/product/emotiv-insight-5-channel-mobile-eeg/

Description – Dry electrode headset allowing the acquisition of 5-channel electroencephalography (EEG) measurements on the AF3, AF4, T7, T8, Pz locations (plus two reference sensors on the left mastoid process). EEG sampling rates are on 128Hz per channel. The device is intended for portability (battery powered and wireless) and field research, rather than for diagnosis or treatment of medical conditions. It also integrates inertial sensors (3 axis accelerometer, gyroscope, and magnetometer) that can acquire data simultaneously with EEG sensors (at 64Hz). The device is provided with API and code for mental command recognition, stress-arousal estimation, and facial expression recognition, which can be used as a baseline for TEACHING PEC estimation algorithms. Collected raw data are streamed through Bluetooth connectivity.

5.2.2.6 Sound

Device - ReSpeaker Mic Array v2.0

Link - www.seeedstudio.com/ReSpeaker-Mic-Array-v2-0.html

Description – Far-field microphone array device capable allowing detection of voices up to a distance of 5 meters. The board integrates 4 high performance digital microphones and an XMOS XVF-3000 processing unit running on-board DSP algorithms for acoustic echo cancellation, beamforming, dereverberation, noise suppression and gain control. The board is integrated into a casing with USB interface for connectivity, control and powering; it also integrates an analogic stereo output. The device is designed for applications such as voice capture, intelligent voice assistant systems, speech-based human-robot interaction, car voice assistant.

5.3 Software tools

5.3.1 Resource orchestration

Kubernetes[10] is the technology of preference in all sorts of solutions when it comes to providing a homogenized execution environment with orchestration capabilities. Along with sibling tools such as K3S[11], Kubernetes can orchestrate the resources of a multitude of infrastructures, as long as they can run containerized software on top of *docker* or *containerd*. The latter resolves

[10] https://kubernetes.io
[11] https://k3s.io

interoperability issues whereas Kubernetes deal with NFRs such as scalability, security, availability, reliability (as parts of dependability), etc.

Furthermore, there are some works in the literature (indicatively: [46], [47]) showing the effectiveness of Kubernetes in providing energy efficiency, however they are all tailored to the specificities of cloud computing and cluster infrastructures.

Regarding safety, Kubernetes maintains some internal processes to ensure that safety at the compute level is achieved, e.g. every non-faulty container executes requests it receives in the same relative order [48].

5.3.2 ML/DL Libraries

ML and DL applications are the basic step towards achieving autonomicity. Thus, the use of ML/DL libraries to support such applications is meant to primarily cover the adaptivity NFR. The most popular deep learning framework is TensorFlow.

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks [49]. Tensorflow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google [50]. It was developed by the Google Brain team for internal Google use and was released under the Apache License 2.0 in 2015. Its closed-source predecessor is called DistBelief.

TensorFlow natively resolves multiple problems at the level of infrastructure, homogenizing and hiding any complexity of the underlying resources for the developer [50]. The security concerns revolving around ML/DL models are many, especially on the topic of adversarial attacks [51]. A large body of literature refers to approaches to deal with those while using TensorFlow. Examples are techniques for secure inference [52], execution of TensorFlow code in a Trusted-Execution Environment [53], etc.

Regarding safety applications, TensorFlow has been widely proposed for safety-critical implementations [54] as well as for techniques for testing, e.g. fault injection [55]. A similar approach is taken in terms of energy efficiency, with the platform being used and tested extensively for its capacity to implement ML/DL applications in an energy-efficient way [56]–[58].

What makes TensorFlow rather appropriate for the needs of TEACHING, besides the familiarity of the researchers and practitioners with it, is also the existence of a lightweight version called TensorFlow Lite[12]. This version provides a better-balanced tradeoff of the TEACHING CPSoS NFRs.

5.3.3 IoT libraries

Supporting IoT devices either for communication purposes or for deploying parts of the applications is crucial for the needs of the TEACHING applications. There are several software libraries for that purpose. For instance, the freeRTOS project provides access to some tools for IoT device shadowing, performance and security metrics monitoring, etc[13].

However, the dominant specification that needs to be supported in TEACHING is the one drafted by OneM2M[14]. The OneM2M technical specifications *address the need for a common*

[12] https://www.tensorflow.org/lite/
[13] https://www.freertos.org/iot-libraries.html
[14] https://onem2m.org/technical/published-drafts/

Machine-to-Machine (M2M) Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide. NFRs such as interoperability, security, safety, reliability and energy-efficiency are dealt with natively in OneM2M [59].

Towards that end, we may focus our attention to the OS-IoT library[15]. This library provides device-side (i.e., Application Entity in oneM2M terminology) support for fundamental oneM2M defined functions. The OS-IoT library provides support for the oneM2M network and protocol functions allowing application developers to interact with the system over a resource-oriented API. By using the library application developers reduce the effort needed to support IoT devices that hook into the oneM2M ecosystem. Instead of having to deal with networks and protocols application developers are freed to focus on the unique, value-added aspects of their application.

5.3.4 Security Libraries

Secure communications and cryptography support is crucial for the TEACHING CPSoS applications. We sought support in native COTS libraries so as to enable the development of secure TEACHING CPSoS applications. The mainstream approach is OpenSSL[16] support. OpenSSL is licensed under an Apache-style license and it is commonly integrated in multiple other software libraries. Therefore, it can be assumed that the appropriate libraries (libssl) are already included in tools like TensorFlow or OS-IoT.

At a higher level, TEACHING may provide complete toolkits with security tools for monitoring and diagnosing security vulnerabilities. Perhaps the most comprehensive such toolkit is the Network Security Toolkit (NST)[17]. Alternatives can be sought to OWASP-branded auditing tools like OWASP ASST[18].

5.3.5 DB Libraries

For storage purposes, the developers of the TEACHING CPSoS applications may benefit from a large array of solutions. Assuming that the tradeoffs between the various NFRs may be similar to that of the Android OS, the obvious solution is SQLite[19]. SQLite implements a small, self-contained SQL database engine. The SQLite file format is stable, cross-platform, and backwards compatible. Among the shortcomings of SQLite the one that is of the highest importance to TEACHING is the security aspects. Security is indeed low in the list of NFRs that the project developers need to meet. There are numerous proposals to resolve various shortcomings (e.g. [60]–[62]) however they all take a significant toll in meeting the rest of the NFRs.

The alternative embedded db system that gains ground nowadays is Oracle Berkeley DB[20]. Berkeley DB is not a relational DB but rathen a document-based, object oriented DB. In other words, if the data are grouped in different tables and there is the need to constantly query result sets that require relating data from more than one table, then the performance will deteriorate significantly. Berkeley DB is better suited for applications using look up tables, i.e., the data is organized in a few tables and there is no need to query data from more than one of them in order

[15] https://os-iot.org/
[16] https://www.openssl.org/
[17] https://www.networksecuritytoolkit.org/nst/index.html
[18] https://github.com/OWASP/ASST
[19] https://www.sqlite.org/index.html
[20] https://www.oracle.com/database/technologies/related/berkeleydb.html

to produce the desired result sets. Berkeley DB is generally very fast and supports a lot of security features, but it will require more work on the application developers' end in order to get the most out of it.

6 TEACHING Platform conceptual architecture

We iterate that the term "TEACHING Platform" is defined as the combined stack of the computing platform and software toolkit upon which a developer develops and deploys CPSoS applications. Figure 26 depicts the TEACHING platform, that encompasses all the notions mentioned in the TEACHING goal and addresses all the requirements. It also comprises the starting point for the design of the individual TEACHING artefacts.

The 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 provided in Section 3.3, 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 as presented in Section 3.2.

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 as proposed in Section 5.

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

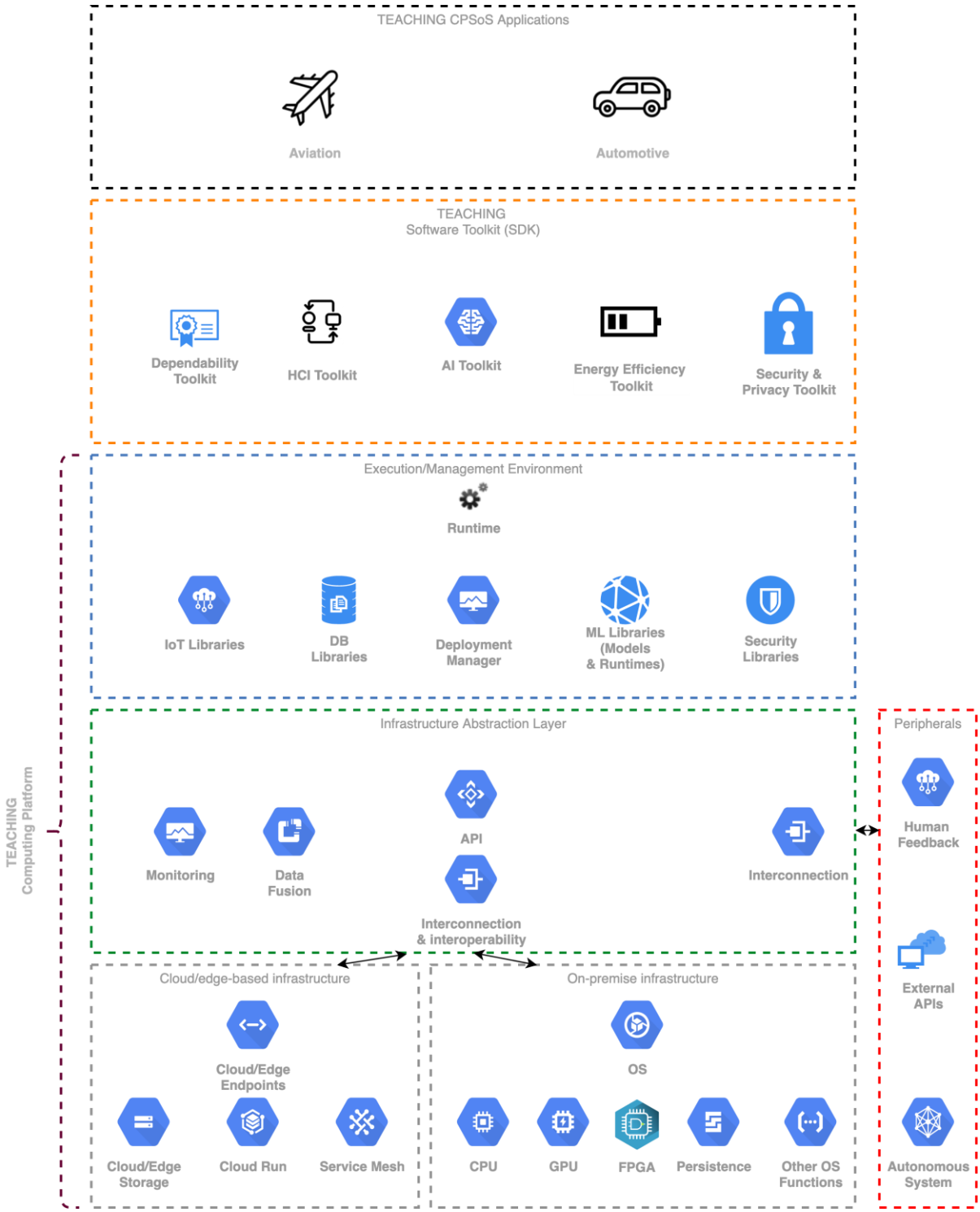

Figure 26: TEACHING platform

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 human centered 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 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 year 1 were 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.

This report provides an overview of work conducted mainly in Tasks 1.1 and 1.3 during this first year of the project towards the implementation of those objectives. The main outcomes presented here are the:

- **Delineation of the scope of the TEACHING CPSoS**: The work defined the TEACHING CPSoS, setting a theoretical frame to the R&D work to be conducted by the technical WPs of the project.
- **Definition of baseline tools and technologies**: Each of the presented tools meets the challenges for the NFRs and can be used for the implementation of the TEACHING Platform. It is a shortlist to be used by the technical WPs while taking into considerations the tradeoffs in meeting the non-functional requirements. The preliminary indications for such tools are:
    - Hardware Platform: *I&M SDF* for the automotive and *iMX8* for the avionics use case
    - Sensors for software monitoring: *METrICS*
    - Sensors for human monitoring: *Shimmer array*, plus *emotive* and *respeaker* sensors
    - Resource orchestration and execution environment: *Kubernetes/K3S*
    - ML/DL libraries/toolkits: *TensorFlow/Lite*
    - IoT libraries/toolkits: *OS-IoT*
    - Security libraries/toolkits: *OpenSSL* (low level), *NST*
    - DB libraries/toolkits: *SQLite*
- **Conceptual architecture of the TEACHING Platform**: This work is expected to facilitate the communication of the framework set in WP1 with the rest of the WPs and act as a precursor to the detailed architecture that is to be released at the end of the second year of the project with D1.2

Those outcomes form a framework for the various versions of requirements elicited at various levels of detail within the technical WPs and as a bullet-tracer for the research and development work to be conducted. As such, we infer that the objectives of WP1 are considered to be achieved.

8 References

[1] A. Platzer, *Logical Foundations of Cyber-Physical Systems*. Springer International Publishing, 2018.

[2] B. H. C. Cheng *et al.*, "Software Engineering for Self-Adaptive Systems: A Research Roadmap," in *Software Engineering for Self-Adaptive Systems*, B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Berlin, Heidelberg: Springer, 2009, pp. 1–26.

[3] H. Müller and N. Villegas, "Runtime Evolution of Highly Dynamic Software," in *Evolving Software Systems*, T. Mens, A. Serebrenik, and A. Cleve, Eds. Berlin, Heidelberg: Springer, 2014, pp. 229–264.

[4] V. E. Silva Souza, A. Lapouchnian, and J. Mylopoulos, "System Identification for Adaptive Software Systems: A Requirements Engineering Perspective," in *Conceptual Modeling – ER 2011*, Berlin, Heidelberg, 2011, pp. 346–361, doi: 10.1007/978-3-642-24606-7_26.

[5] M. Abbaszadeh and R. Solgi, "Constrained Nonlinear Model Predictive Control of an MMA Polymerization Process via Evolutionary Optimization," *ArXiv150204266 Cs Math*, Feb. 2015, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1502.04266.

[6] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, "Approximate model predictive building control via machine learning," *Appl. Energy*, vol. 218, pp. 199–216, May 2018, doi: 10.1016/j.apenergy.2018.02.156.

[7] D. Görges, "Relations between Model Predictive Control and Reinforcement Learning," *IFAC-Pap.*, vol. 50, no. 1, pp. 4920–4928, Jul. 2017, doi: 10.1016/j.ifacol.2017.08.747.

[8] A. Avizienis, J.- Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004, doi: 10.1109/TDSC.2004.2.

[9] L. Zhang, Q. Wang, and B. Tian, "Security threats and measures for the cyber-physical systems," *J. China Univ. Posts Telecommun.*, vol. 20, pp. 25–29, Aug. 2013, doi: 10.1016/S1005-8885(13)60254-X.

[10] T. Lu, J. Zhao, L. Zhao, Y. Li, and X. Zhang, "Towards a Framework for Assuring Cyber Physical System Security," *Int. J. Secur. Its Appl.*, vol. 9, pp. 25–40, Mar. 2015, doi: 10.14257/ijsia.2015.9.3.04.

[11] E. K. Wang, Y. Ye, X. Xu, S. M. Yiu, L. C. K. Hui, and K. P. Chow, "Security Issues and Challenges for Cyber Physical System," in *2010 IEEE/ACM Int'l Conference on Green Computing and Communications Int'l Conference on Cyber, Physical and Social Computing*, Dec. 2010, pp. 733–738, doi: 10.1109/GreenCom-CPSCom.2010.36.

[12] S. Adepu *et al.*, "Control Behavior Integrity for Distributed Cyber-Physical Systems," *ArXiv181208310 Cs*, Dec. 2018, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1812.08310.

[13] "Establishing Data Integrity in Networks of Cyber-Physical Systems - IEEE Conference Publication." https://ieeexplore.ieee.org/document/8550617 (accessed Dec. 17, 2020).

[14] H. Peng, C. Liu, D. Zhao, H. Ye, Z. Fang, and W. Wang, "Security Analysis of CPS Systems Under Different Swapping Strategies in IoT Environments," *IEEE Access*, vol. 8, pp. 63567–63576, 2020, doi: 10.1109/ACCESS.2020.2983335.

[15] M. Rungger and P. Tabuada, "A Notion of Robustness for Cyber-Physical Systems," *IEEE Trans. Autom. Control*, vol. 61, no. 8, pp. 2108–2123, Aug. 2016, doi: 10.1109/TAC.2015.2492438.

[16] M. Lokesh, Y. Kumaraswamy, and K. Tejaswini, “Challenges and Current Solutions of Cyber Physical Systems,” *IOSR J. Comput. Eng.*, vol. 18, pp. 2278–661, Mar. 2016, doi: 10.9790/0661-1821104110.
[17] “Multi-cyber framework for availability enhancement of cyber physical systems | SpringerLink.” https://link.springer.com/article/10.1007/s00607-012-0227-7 (accessed Dec. 17, 2020).
[18] L. Miclea and T. Sanislav, “About dependability in cyber-physical systems,” in *2011 9th East-West Design Test Symposium (EWDTS)*, Sep. 2011, pp. 17–21, doi: 10.1109/EWDTS.2011.6116428.
[19] L. Vegh and L. Miclea, “Enhancing security in cyber-physical systems through cryptographic and steganographic techniques,” in *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, May 2014, pp. 1–6, doi: 10.1109/AQTR.2014.6857845.
[20] W. Jiang, L. Wen, J. Zhan, and K. Jiang, “Design optimization of confidentiality-critical cyber physical systems with fault detection,” *J. Syst. Archit.*, vol. 107, p. 101739, Aug. 2020, doi: 10.1016/j.sysarc.2020.101739.
[21] K. Jiang, P. Eles, and Z. Peng, “Co-design techniques for distributed real-time embedded systems with communication security constraints,” in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2012, pp. 947–952, doi: 10.1109/DATE.2012.6176633.
[22] A. Haqiq, *Cybersecurity and Privacy in Cyber-Physical Systems*. CRC Press, 2019.
[23] E. B. Fernandez, “Threat Modeling in Cyber-Physical Systems,” in *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, Aug. 2016, pp. 448–453, doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2016.89.
[24] H. Neema, B. Potteiger, X. Koutsoukos, G. Karsai, P. Volgyesi, and J. Sztipanovits, “Integrated simulation testbed for security and resilience of CPS,” in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, New York, NY, USA, Apr. 2018, pp. 368–374, doi: 10.1145/3167132.3167173.
[25] M. Burmester, E. Magkos, and V. Chrissikopoulos, “Modeling security in cyber–physical systems,” *Int. J. Crit. Infrastruct. Prot.*, vol. 5, no. 3, pp. 118–126, Dec. 2012, doi: 10.1016/j.ijcip.2012.08.002.
[26] E. R. Griffor, “Framework for Cyber-Physical Systems: Volume 1, Overview,” *NIST Spec. Publ.*, vol. 1, p. 79.
[27] “Good Practices for Security of Internet of Things in the context of Smart Manufacturing.” https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot (accessed Dec. 17, 2020).
[28] R. Conradi and T. Lauritsen, “Mini-glossary of software quality terms, with emphasis on safety,” NTNU, Jun. 2007. Accessed: Dec. 17, 2020. [Online]. Available: https://studylib.net/doc/9025338/mini-glossary-of-software-quality-terms--with-emphasis-on...
[29] S. Nadjm-Tehrani, “Functional safety and IEC 61508: A basic guide,” International Electrotechnical Commission, May 2004. Accessed: Dec. 17, 2020. [Online]. Available: https://www.ida.liu.se/~simna73/teaching/SCRTS/IEC61508_Guide.pdf.
[30] A. Specification, *653-2,“Avionics Application Software Standard Interface,” December 1, 2005*. .
[31] P. Parkinson and L. Kinnan, “Safety-critical software development for integrated modular avionics,” *Embed. Syst. Eng.*, vol. 11, no. 7, pp. 40–41, 2003.

[32] *DO-178B: Software considerations in airborne systems and equipment certification*. RTCA, Incorporated, 1992.
[33] "DO-254: Hardware considerations in airborne systems and equipment certification," Radio Technical Commission for Aeronautics & EURopean Organisation for Civil Aviation Equipment, 1992.
[34] 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.
[35] E. Mezzetti and T. Vardanega, *On the industrial fitness of wcet analysis*. na, 2011.
[36] 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.
[37] 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.
[38] J. Bin, S. Girbal, D. Gracia Pérez, A. Grasset, and A. Mérigot, "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.
[39] 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.
[40] S. Girbal, X. Jean, J. L. Rhun, D. G. Pérez, and M. Gatti, "Deterministic platform software for hard real-time systems using multi-core COTS," in *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, Sep. 2015, pp. 8D4-1-8D4-15, doi: 10.1109/DASC.2015.7311481.
[41] S. Mittal, "A survey of techniques for improving energy efficiency in embedded computing systems," *Int. J. Comput. Aided Eng. Technol.*, vol. 6, no. 4, p. 440, 2014, doi: 10.1504/IJCAET.2014.065419.
[42] L. Liu *et al.*, "Computing Systems for Autonomous Driving: State-of-the-Art and Challenges," *ArXiv200914349 Cs*, Dec. 2020, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/2009.14349.
[43] B. Sprunt, "The basics of performance-monitoring hardware," *IEEE Micro*, vol. 22, no. 4, pp. 64–71, Jul. 2002, doi: 10.1109/MM.2002.1028477.
[44] E. Duesterwald, C. Cascaval, and Sandhya Dwarkadas, "Characterizing and predicting program behavior and its variability," in *2003 12th International Conference on Parallel Architectures and Compilation Techniques*, Sep. 2003, pp. 220–231, doi: 10.1109/PACT.2003.1238018.
[45] S. Girbal, J. L. Rhun, and H. Saoud, "METrICS: a Measurement Environment For Multi-Core Time Critical Systems," presented at the ERTS 2018, Jan. 2018, Accessed: Dec. 17, 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02278292.
[46] K. Kaur, S. Garg, G. Kaddoum, S. H. Ahmed, and M. Atiquzzaman, "KEIDS: Kubernetes-Based Energy and Interference Driven Scheduler for Industrial IoT in Edge-Cloud Ecosystem," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4228–4237, May 2020, doi: 10.1109/JIOT.2019.2939534.
[47] P. Townend *et al.*, "Invited Paper: Improving Data Center Efficiency Through Holistic Scheduling In Kubernetes," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, Apr. 2019, pp. 156–15610, doi: 10.1109/SOSE.2019.00030.

[48] H. V. Netto, L. C. Lung, M. Correia, A. F. Luiz, and L. M. Sá de Souza, "State machine replication in containers managed by Kubernetes," *J. Syst. Archit.*, vol. 73, pp. 53–59, Feb. 2017, doi: 10.1016/j.sysarc.2016.12.007.
[49] M. Abadi *et al.*, "TensorFlow: A System for Large-Scale Machine Learning," 2016, pp. 265–283, Accessed: Dec. 20, 2020. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.
[50] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *ArXiv160304467 Cs*, Mar. 2016, Accessed: Dec. 20, 2020. [Online]. Available: http://arxiv.org/abs/1603.04467.
[51] N. Akhtar and A. Mian, "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018, doi: 10.1109/ACCESS.2018.2807385.
[52] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow: Secure TensorFlow Inference," in *2020 IEEE Symposium on Security and Privacy (SP)*, May 2020, pp. 336–353, doi: 10.1109/SP40000.2020.00092.
[53] R. Kunkel, D. L. Quoc, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer, "TensorSCONE: A Secure TensorFlow Framework using Intel SGX," *ArXiv190204413 Cs*, Feb. 2019, Accessed: Dec. 20, 2020. [Online]. Available: http://arxiv.org/abs/1902.04413.
[54] C.-H. Hsieh, D.-C. Lin, C.-J. Wang, Z.-T. Chen, and J.-J. Liaw, "Real-Time Car Detection and Driving Safety Alarm System With Google Tensorflow Object Detection API," in *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, Jul. 2019, pp. 1–4, doi: 10.1109/ICMLC48188.2019.8949265.
[55] G. Li, K. Pattabiraman, and N. DeBardeleben, "TensorFI: A Configurable Fault Injector for TensorFlow Applications," in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Oct. 2018, pp. 313–320, doi: 10.1109/ISSREW.2018.00024.
[56] H. Jo and Y. I. Yoon, "Intelligent smart home energy efficiency model using artificial TensorFlow engine," *Hum.-Centric Comput. Inf. Sci.*, vol. 8, no. 1, p. 9, Apr. 2018, doi: 10.1186/s13673-018-0132-y.
[57] M. Alzantot, Y. Wang, Z. Ren, and M. B. Srivastava, "RSTensorFlow: GPU Enabled TensorFlow for Deep Learning on Commodity Android Devices," in *Proceedings of the 1st International Workshop on Deep Learning for Mobile Systems and Applications*, New York, NY, USA, Jun. 2017, pp. 7–12, doi: 10.1145/3089801.3089805.
[58] D. Li, X. Chen, M. Becchi, and Z. Zong, "Evaluating the Energy Efficiency of Deep Convolutional Neural Networks on CPUs and GPUs," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, Oct. 2016, pp. 477–484, doi: 10.1109/BDCloud-SocialCom-SustainCom.2016.76.
[59] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common M2M service layer platform: Introduction to oneM2M," *IEEE Wirel. Commun.*, vol. 21, no. 3, pp. 20–26, Jun. 2014, doi: 10.1109/MWC.2014.6845045.
[60] H. Liu and Y. Gong, "Analysis and design on security of sqlite," 2013.
[61] S. Mutti, E. Bacis, and S. Paraboschi, "Sesqlite: Security enhanced sqlite: Mandatory access control for android databases," in *Proceedings of the 31st Annual Computer Security Applications Conference*, 2015, pp. 411–420.
[62] Y. Wang, Y. Shen, C. Su, J. Ma, L. Liu, and X. Dong, "CryptSQLite: SQLite With High Data Security," *IEEE Trans. Comput.*, vol. 69, no. 5, pp. 666–678, 2019.