Safe and Explainable
Critical Embedded Systems based on AI

D1.1 Requirements, Success Criteria and Platforms

Version 1.0

Documentation Information

Contract Number	101069595
Project Website	www.safexplain.eu
Contratual Deadline	30.03.2023
Dissemination Level	PU
Nature	R
Author	Francisco J. Cazorla (BSC), Irune Agirre (IKR), Fahad Sarfraz (NAV), Thanh Bui (RISE)
Contributors	Jaume Abella (BSC), Enrico Mezzetti (BSC), Ane Mijangos (IKR), Javier Fernandez (IKR)
Reviewer	Gabriele Giordana (AIKO)
Keywords	Requirements, Success Criteria, Platform

This project has received funding from the European Union's Horizon Europe programme under grant agreement number 101069595.

Change Log

Version	Description Change
V0.1	First draft
V0.2	Comments from reviewer
V1.0	Final Version

Table of Contents

Executive Summary

During the first 6 months of the project, in this Work Package (WP) the team has focused on the identification and validation of the requirements related to the main areas of research in the project (FUSA, DL, and platforms), and on the definition of the success criteria to be achieved in each milestone (T1.1). Also, effort has been devoted to the selection and setup of the baseline hardware platforms on which case studies will be run as well as the identification of the baseline SAFEXPLAIN software stack (T1.2). During these first 6 months, we have also defined an early set of safety requirements and patterns to steer WP2, WP3, WP4 starting in m4 (T1.3).

This document therefore captures: the case study and FUSA standards requirements, the success criteria set for each milestone of the project, the result of the selection of the platform and software stack, and FUSA principles to consider in SAFEXPLAIN.

1. Requirements and Success Criteria

At the top level, D1.1 captures the following types of requirements.

- Safety Assessment (SA): this requirements set relates to the considerations that shall be taken into account in SAFEXPLAIN when addressing the challenges associated to functional safety. Requirements cover the scope of safety assurance case templates, lifecycle considerations, safety analyses, architecture designs and certification-related activities.
- Deep Learning (DL): the requirements are focusing on (i) compliance with safety requirements, including specificability and evaluation metrics; and (ii) explainability related requirements for DL components including: representation types, architecture, and feasibility of implementation.
- Tool Set Support and Platform: under this umbrella we consider all technological requirements to support SAFEXPLAIN goals in terms of supported functionalities, time predictability, and mixed-criticality execution. Requirements cover several aspects related to performance, libraries and toolchain support, platform observability, controllability, monitoring of relevant metrics, as well as functional and non-functional software verification in general.
- Case studies: these are end-user type of requirements to ensure that the case studies can be integrated with the software stack developed in the other main lines of the project (SA, DL, and tool set support) on the target platform so as to assess the benefits of the project. The case study requirements also ensure that the explanations and metrics developed in other WPs are aligned with and applicable to the use cases throughout their respective development.
- Platform selection: Underpinning all these are the requirements on the platform that should achieve the requirements set by the different lines of the project. This is specifically covered in Section 2.

In each of these main lines the requirements were broken down into subcategories as follows.

1.1. Requirements

1.1.1. Safety Assessment (WP2)

Requirements related with the safety assessment (WP2) determine the scope of the activities that the project will perform towards the safety certifiability of Machine Learning (ML)-based solutions. To this end, the following aspects are considered:

- Standards. Safety activities of the project shall be based on existing emerging standards. As the spectrum of available and emerging standards in the topic is very wide, this requirement relates to the selection of a set of reference standards, which is covered in Section 3.1 of this report.
- Safety assurance cases. A safety case defines the arguments and evidence required to demonstrate a given goal (i.e., that a DL subsystem is sufficiently safe for its purpose) in a structured way. These requirements cover the scope of the safety cases that will be produced as an outcome of SAFEXPLAIN.
- Safety architectural patterns: These requirements cover the aspects that shall be considered when designing safe system architectures including ML components. These architecture patterns will be based on mechanisms such as redundancy, diversity,

diagnostic mechanisms and supervision elements later introduced in Section 3.2. Requirements specify the needs in terms of residual risk calculation, adversarial attacks and platforms on which the architectures templates shall be tailored.
- Safety assessment and expert reviews: Requirements for the final assessment of WP2 outcomes through external expert reviews are addressed in this last group.

1.1.2. Deep Learning (WP3)

The requirements for WP3 belong to the following main categories:

- Alignment with Safety assessment: The alignment with WP2 is critical and should be performed in interactive mode to maintain the agreements in terms of common definitions, compliance means, and architecture.
- Addressing DL uncertainties: The explainability should address both data and DL model.
- Addressing V&V requirements: The extracted explanations should focus on different audiences such as DL developer and safety experts. The information should be represented in agreed forms to support safety argumentations.
- Feasibility: The explainable AI methods should leverage the latest developments of Service Oriented Architecture (SoA) methods, with focus on feasibility to implement on agreed platform (WP4) and with DL components to be used in UCs.

1.1.3. Tool Set Support and Platform (WP4)

The objectives on the tool set support and platform (WP4) have been broken down into two main areas: (i) the hardware platform itself together with the available libraries and tool-chain support, and (ii) the support for SAFEXPLAIN objectives in terms of DL explainability, verification, and safe mixed criticality execution.

The first area relates to the selection of a hardware platform and software environment on which to develop SAFEXPLAIN solutions. These elements are not meant to be developed in the scope of the project and hence whose design, development, and implementation cannot be affected. The work done in WP1 has thus focused on identifying and selecting the hardware platform and software environment that better fits the needs of the project under a reasonable budget. We report on the selection criteria and results in Section 2.

The second area, instead, relates to the set of techniques and supporting tools that are meant to be designed and developed within the scope of the project and we will develop to capture the project objectives and, ultimately, to support the subset of safety requirements addressed in the project. We distinguish the following main scopes:

- Timing characterization. This covers the requirements related to the software timing analysis part of the project, mainly related to the analysis techniques and tools to be deployed for the timing analysis of the complex DL functionalities supported by the SAFEXPLAIN framework.
- Observability and controllability. These requirements relate to exploiting the main means for collecting low-level information on the execution of DL applications on the target platform (via hardware events) and on the control over Quality of Service (QoS) and hardware configuration to support FUSA and mixed-criticality objectives.
- DL Libraries integration on Validation toolset. This set of requirements is addressing the development of a partially automated framework for the execution and verification of the

DL functionalities in the use cases by providing flexible monitoring functionalities to gather relevant metrics, including those specifically identified in SAFEXPLAIN. All in all, this ensures DL libraries used and possible adapted in the project properly execute on top of the SAFEXPLAIN validation toolset.

1.1.4. Case Studies (WP5)

The requirements around the case studies covered two main lines:

- Requirements on the case studies. These cover the stubbing and changes required in the case studies to run on the selected hardware and software platform. This also involves the creation and specification of training datasets to train the required DL components, specifications and the performance requirements of the DL models and finally the test cases to cover and validate the stipulated safety scenarios. Requirements from other WP's were collected to coordinate the respective work, such as identifying the case studies safety scenarios, the algorithms architectures and some early simplified models to test the software stack prototype.
- Requirements from the case studies. These requirements capture needs emanating from the case studies towards the rest of the software stack. They include the hardware and software requirements on the platform and the relevant DL components (explanations and metrics) from WP3 which are required to meet the safety requirements. Some entries explicate technical needs, such as the availability on the platform of resources such as accelerators and the compatibility with different DL frameworks. Others ensure that incremental prototypes from other WP's work, and related requirements which may arise, are provided at major milestones so that to monitor and enforce a common direction throughout the project.

1.2. Success Criteria

There are 4 main control points or Milestones in the project: MS1 (Requirements, success criteria and platforms selection), MS2 (Concept development, technology prototyping, and case studies preliminary porting), MS3 (Technology consolidation and case studies integration), MS4 (Final Assessment).

For each requirement we define a success criterion in at least one of the four Milestones. The goal is to "land" each requirement into a specific point in time in the project in which we assess the total fulfilment of the requirement.

For some requirements we analyse its partial fulfilment in different milestones, the last of which assess the competition of the requirement.

1.3. Summary

Attachment 1 (REQs&SC.pdf) provides a view of all the requirements identified as well as the milestones in which its fulfilment is partially or totally assessed.

2. Platform Selection and Setup

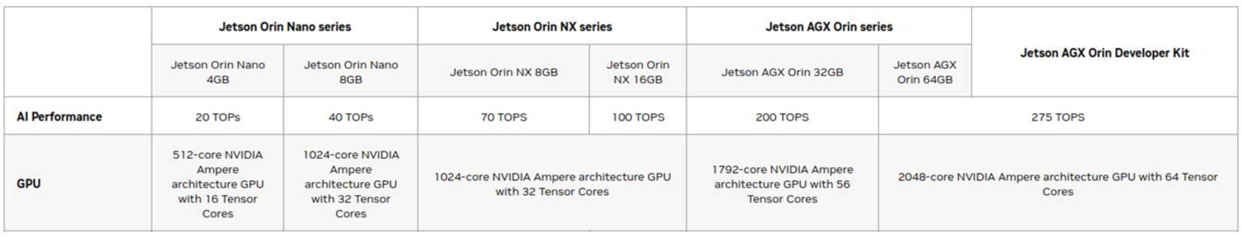
2.1. Hardware Platform

During the kick-off meeting of the project, and the weeks following it, we reviewed the case study requirements in terms of sustained performance and library support. During the KoM, all project partners agreed on the convenience of converging to one common target board across use cases. This would have allowed narrowing the scope of the implementation and demonstration activities without restricting the scope and validity of the project results.

In order to facilitate the selection process, the case study providers were asked to express their initial requirements on the Hardware (HW)/ Software (SW) stack. The result of this activity are summarized in Table 1. From the analysis of those requirements, the Jetson AGX Orin series (together with the respective SW stack) has been considered by all partners and eventually identified as a good candidate for the final selection.

Among the three main variants of the Orin, the Jetson AGX Orin Development Kit have been considered as a promising candidate from a performance perspective since, based on the information available on NVIDIA website, it provides comparable performance to the powerful AGX Orin 64GB module (https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/) with the advantage of an almost ready-to-use setup, with a consolidated Software Development Kit (SDK) and supporting all main accelerator libraries. It is also possible to purchase the 64GB module but in that case we also need to procure cable, power unit, and a carrier board. A summary table from NVIDIA website is reported below.

Table 1. AGX Orin summary comparative chart (form NVIDIA website).



	Jetson Orin Nano series		**Jetson Orin NX series**		**Jetson AGX Orin series**		**Jetson AGX Orin Developer Kit**
	Jetson Orin Nano 4GB	Jetson Orin Nano 8GB	Jetson Orin NX 8GB	Jetson Orin NX 16GB	Jetson AGX Orin 32GB	Jetson AGX Orin 64GB	
AI Performance	20 TOPs	40 TOPs	70 TOPS	100 TOPS	200 TOPS	275 TOPS	
GPU	512-core NVIDIA Ampere architecture GPU with 16 Tensor Cores	1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores	1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores		1792-core NVIDIA Ampere architecture GPU with 56 Tensor Cores	2048-core NVIDIA Ampere architecture GPU with 64 Tensor Cores	

After looking into procurement times for the Jetson AGX Orin Development Kit, we observed a reasonable approx. 8 weeks lead time. The price of the board was also reasonable, landing at ~2K/2.5K Euros + VAT.

2.2. Software Stack

Once the board was agreed among partners, the next step was to converge on a software stack that allows all partners to run their case study while allowing the technical WPs to carry out their intended work. Table 3 captures the requirements identified by each case study on the different element of the software stack.

The main goal of the SW stack selection was to identify the exact version of support libraries and tool-chain to be fixed as part of SAFEXPLAIN software development environment. By doing this we wanted to avoid late issues from incompatibilities among software packages. As final step, partners agreed in a version of all the software packages that satisfy the requirements of all the case studies. Those versions are shown in Table 4.

Table 2. Case study providers have expressed their initial requirements on the HW/SW stack as summarized below.

Partner	Domain	Short description	Required HW features	HW Prospects	SW Components and Libraries	Programming Languages	SW Environment	Data	Perf req.
AIK	Space	Autonomous navigation. Image acquisition for estimating the spacecraft position	Nvidia GPU	Jetson AGX Orin Series?	TensorFlow	C++	Std Linux Env.	Camera images + sensor time series (if possible)	To be assessed
			Xilinx FPGA		PyTorch	Python	In-house automated framework		
			Intel Myriad VPU		Platform Inference Engine				
			Google Edge TPU [others]						
NAV	Auto	Pedestrian detection	NVMe storage	Jetson AGX Xavier 64GB	Jetson JetPack 5.0<	Python 3.7<	Std Linux Env.	Training dataset	20-30 FPS
			ARM CPU	Jetson AGX Xavier 32GB					
			NVIDIA GPU	Jetson AGX Xavier Industrial					
			Webcam/Dashcam/ other	Jetson AGX Orin 32GB					
				Jetson AGX Orin 64GB					
IKR	Railway	Automatic Train Operation (ATO) Level 4 Fully autonomous. Detect and localization of obstacles on tracks + distance estimation		Jetson AGX Orin	Yolo	C		Public Dataset (from Auto)	To be assessed
					Stereo Vision with OpenCV API			Real field images (not sure)	
					TensorFlow AI framework			Train simulator	
					Proprietary C Lib				
					Jetson SW Stack				

Table 3. Requirements each of the SAFEXPLAIN case studies put on the different elements of the software stack.

	automotive	**railway**	**space**	**conclusion**
Operating System	Ubuntu 18.4 and above, preferably 20.4			Ubuntu 18.4 and above, preferably 20.4 kernel linux 5.10>
AI top-level libraries	Preferably PyTorch 1.12 and above	PyTorch and Tensorflow+keras, we want both options to experiment.	use case on pytorch; can be imported as ONNX and used with tensorRT (issues for integration with explainability? especially during training)	pytorch (main framework) 1.12> tensor RT 8.4> optional,
Performance-improving libraries	Jetpack 5.0.2, TensorRT v8.4.1	Jetpack, TensorRT	JetPack SDK latest version (v5), use of GPU with tensorRT and Nvidia accelerators	JetPack 5.X
Application software	Apollo	OpenCV 4.4>	OpenCV, python libraries	Cuda 11.4> OpenC 4.4> python 3.8 or 3.9 supporting Apollo v4 or v7

Table 4. Version of each software package agreed by every parnter in the project.

Software	**Version**
Jetpack	5.1
Jetson Linux	35.2.1
Ubuntu Version	20.04
Kernel Version	5.10.65-tegra
Tensor RT	8.5.2
cuDNN	8.6.0
CUDA	11.4.19
OpenCV	4.5.4
Python	3.8.10
PyTorch	1.14
Vulcan	1.3.203
Vulcan SC	1.0

3. FUSA principles

With the aim of identifying the main functional safety principles that will guide the technical solutions of the project, this section first surveys existing and emerging safety standards, technical reports and guidelines on the three principal disciplines of the project (i.e., functional safety, deep learning and high-performance embedded platforms) for the three industrial domains of the case studies (railway, automotive and space). Based on selected standards, this section then describes early directions and baseline safety principles to start designing safe DL-based solutions. Note that the project will keep checking for new or updates on standards in its next phases.

3.1. Standards

The deployment of safety-critical systems requires to comply with legal regulations in place on each country or state. This often involves a certification process where an independent certification institution must approve, through a conformity assessment, that the system is suitable and safe enough for its intended use (i.e., that it is compliant with the active legal directives). Currently, the most common certification approach is standard based, i.e., a conformity assessment is achieved by proving adherence to the applicable safety standards.

In traditional functional safety approaches, freedom from unacceptable risk is pursued by dealing with systematic and random faults on the system to prevent them from causing destructive effects (e.g., loss of lives or environmental damage). With the advent of autonomous systems, new challenges arise such as addressing, not only the malfunctioning of the system (functional safety), but also the functional insufficiencies or deficiencies of the system that could lead to hazardous situations (also known as Safety of the intended functionality (SOTIF)). Therefore, the normative landscape is also evolving to cover these autonomous systems, often based on Artificial Intelligence (AI) techniques that require high performance computing platforms. Next subsections survey existing and emerging standards in these complementary disciplines and concludes with a selection of most interesting standards to take as reference in the initial phases of the project.

3.1.1. Functional Safety Standards

Functional safety standards, such as, IEC-61508 [1], ISO-26262 [2], EN-5012x [3], define the requirements for the development of safety related electrical and/or electronic systems with the purpose of avoiding unacceptable risks in the system. As illustrated in Figure 1, many of these standards apply to the specific domain for which they have been conceived (e.g., automotive, railway, elevation, avionics, space). However, except for the avionics and space domains, several industrial safety standards use as a reference the generic IEC 61508 [1] safety standard, sharing many similarities.

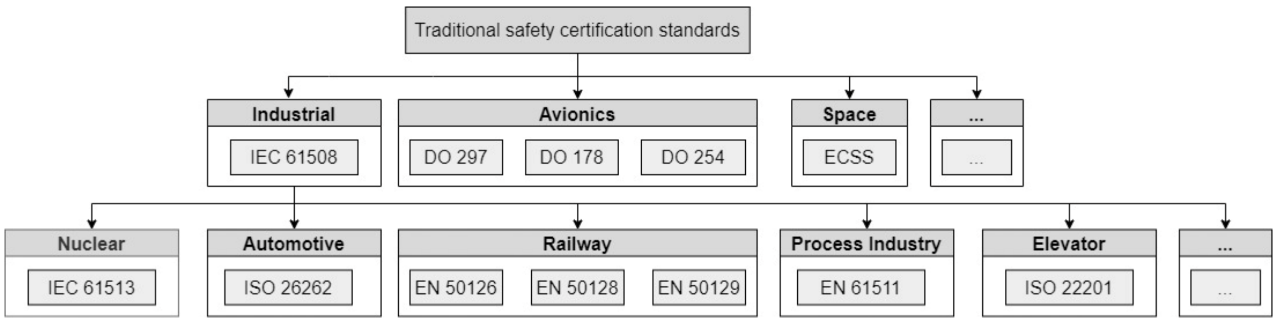


Figure 1: Functional safety standards and their relationship

- **IEC 61508 (Generic), ISO 26262 (Automotive), EN 5012x (Railway)**

As it can be seen in Figure 1, IEC 61508 [1] is a reference generic functional safety standard and many domain specific standards are based on it (e.g., industrial machinery (ISO 13849) [4] robotics (ISO 10218) [5] tractors, machinery for agriculture (ISO 18497 [6]), process industry (EN 61511 [7]),…)). Functional safety standards for Automotive (ISO 26262 [2]) and Railway (EN 50126, EN 50128, EN 50129 [3]) fall into this category.

These standards are based on the prevention of systematic errors (through rigorous process measures throughout the system lifecycle) and control and protection against random errors (e.g., by diagnostic mechanisms that move the system to a safe state). Safety functions are classified by a criticality level (i.e., Safety Integrity Level (SIL), Automotive Safety Integrity Level (ASIL)) based on their severity, frequency of exposure and controllability of the hazardous events. The higher the criticality level, more stringent are the procedures, measures, and requirements of functional safety standards.

When it comes to the use of AI, IEC 61508 edition 2 only mentions artificial intelligence as a fault correction technique in the software architecture design and development (e.g., for fault forecasting, fault correction or maintenance activities supported by artificial intelligence). Despite this limitation of considering it only for fault correction, the standard does not recommend (NR) its usage in systems with an integrity level higher than SIL 1 (Table 5). This requirement may be adapted in future editions of IEC 61508.

Table 5: Extract from IEC 61508-3 (ed 2) – Software design and development – software architecture design

Technique	SIL 1	SIL 2	SIL 3	SIL 4
Artificial Intelligence – Fault correction	---	NR	NR	NR

- **ECSS-Q-ST-30C/ECSS-Q-ST-40C/ECSS-Q-HB-80-03A - Space**

ECSS is a cooperative effort of the European Space Agency (ESA), national space agencies and European industry associations, aiming to develop and maintain common standards for space activities. The listed standards of this section belong to the series of ECSS Standards meant to be applied together for the management, engineering and product assurance in European space projects and applications.

  - **ECSS-Q-ST-30C Rev.1 (Dependability):** This standard [8] defines how to assure the dependability for space systems and the requirements for it. The dependability requirements for functions implemented in software, and the interaction between hardware and software, are identified in this Standard. A classification of these functions in accordance with their criticality is presented.
  - **ECSS-Q-ST-40C Rev.1 (Safety):** This Standard [9] defines the safety program and the safety technical requirements aiming to protect every person/element involved in space activity from flight and ground personnel, the environment and the civilian population, launch vehicles, ground support equipment, private and public property etc… from hazards associated with European space systems. As in the case of previous functional safety standards, risk reduction is pursued by the:
    - identification of all safety related risks with respect to the design, development and operations of space products
    - assessment of the risks based on qualitative and quantitative analysis

        - application of a hazard reduction precedence and of control measures of the residual risks.
    - **ECSS-Q-HB-80-03A Rev.1 (Software dependability and safety):** This standard [10] provides a general description of the entire software dependability and safety workflow, considering the different activities at system and software level, the lifecycle phases and the customer-supplier relationships. Some software Reliability, Availability, Maintainability and Safety (RAMS) techniques are presented. The given techniques are the result of a selection of the existing techniques relevant to the requirements defined in the ECSS Standards.
- **DO-178C – Avionics- Software Considerations in Airborne Systems and Equipment Certification**

DO-178C standard covers the complete software lifecycle (planning, development and integral processes) to guarantee correctness and robustness in software systems for civil airborne applications. The Design Assurance Level (DAL determines the amount of rigor required by the design assurance process. DAL categorization is determined by the impact that the specific system's failure could have in terms of Aircraft Safety. The more critical the DAL, the more activities and objectives are required.

3.1.2. AI and Autonomous Systems Standards

In recent years, as a result of the increasing popularity of autonomous systems, new standards are emerging to cover the safety challenges involved by the increasing level of autonomy and technologies on which they rely, such as artificial intelligence (AI). In this section we survey existing and emerging standards in this field. While the main focus is on safety-related standards, we also mention some related to information technology as they introduce interesting terminology and concepts such as explainability.

- **ISO/IEC TR 5469 - Artificial intelligence – Functional safety and AI systems**

ISO / IEC 5469 standard aims to cover the application of AI-based solutions on safety-critical systems by identifying the properties, safety risk factors, available methods and potential constraints towards the appropriate adoption of AI approaches in safety functions. The standard is not associated to any application domain. At the time of writing, this standard is still at a development phase and the information on this section is based on early drafts.

This standard is of particular interest for AI-based systems development, as it covers different aspects of AI safety functions. For instance, it defines a high-level lifecycle that combines the V-model and ML lifecycle activities, it identifies the properties to be considered and evaluates the potential compliance of AI-based solutions with existing functional safety standards.

On the platform side, the standard identifies the technology elements required for ML model creation and execution and differentiates among those which can be covered by traditional functional safety techniques from those that require further considerations. It also mentions that GPU based systems may have special failure modes to be addressed and some architectural considerations are proposed (like the use of supervisors, redundancy and diversity and detection mechanisms).

- **ISO/IEC 24029 - Artificial intelligence — Assessment of the robustness of neural networks**

Robustness, resiliency, reliability, accuracy, safety, security, privacy are properties seek when designing any system. Among these, robustness is a crucial property. In the context of AI systems, this property brings new challenges as they can sometimes have unexpected behaviour and are hard to explain due to their non-linear nature. Robustness is capital when it comes to validation. In many organizations, software validation is an essential step in order to put the software into production.

The techniques used in AI systems are also subject to validation. However, common techniques used in AI systems require specific approaches in order to ensure adequate testing and validation.

This document [11] provides an overview about the existing methods/approaches to measure the robustness of neural networks.

- **ISO/DIS 21448 - Road vehicles — Safety of the intended functionality**

In ISO 26262 the functional safety is defined as the absence of unreasonable risk due to hazards caused by a malfunctioning behaviour of the electrical/electronical system. But in some electrical/electronical systems relying on the environment to build awareness on the situation, even if those systems are free from malfunctions, there can be hazardous behaviour caused by the intended functionality. This is where the focus of ISO 21448 is, which is known as the safety of the intended functionality (SOTIF). SOTIF is then the absence of unreasonable risk due to a hazard caused by functional insufficiencies such as:

  - the inability of the function to correctly perceive the environment.
  - the lack of robustness of the function, system, or algorithm with respect to sensor input variations, heuristics used for fusion, or diverse environmental conditions.
  - the unexpected behaviour due to decision making algorithm and/or divergent human expectations.

Functional safety (addressed by the ISO 26262 series) and SOTIF are distinct and complementary aspects of safety. This standard provides a general argument framework and guidance on the applicable design, verification and validation measures, as well as activities during the operation phase to ensure the safety of the intended functionality.
ISO 21448 includes an interesting Annex about the implications of Machine Learning, which mentions the need of methods to mitigate ML component performance insufficiencies and to mitigate systematic faults introduced by the training process and their corresponding data collection processes.

- **ISO PAS 8800 - Road Vehicles - Safety and Artificial Intelligence**

This document sets the definition of safety-related properties and risk factors that impact the insufficient performance and malfunctioning behaviour of AI for road vehicles.

It sets a framework that addresses all development phases and life cycle of IA components. This framework takes into consideration the derivation of suitable safety requirements on the function and factors related to data quality and completeness. It provides architectural measures for the control and mitigation of failures and defines tools used to support AI as well as verification and validation techniques. The evidence required to support an assurance argument for the overall safety of the system is additionally described.

The objectives of this standard are the following (see Figure 2):

  - Define suitable safety principles, methods and evidence satisfying objectives with ISO 26262 [2] and ISO 21448 [12]
  - Harmonize concepts described in ISO/TR 4804 [13] and ISO 21448 [12] Annexes'
  - Rely on generic guidance from ISO/IEC TR 5469 [14]

At the time of writing, this standard is still at a development phase and information is based on early drafts.

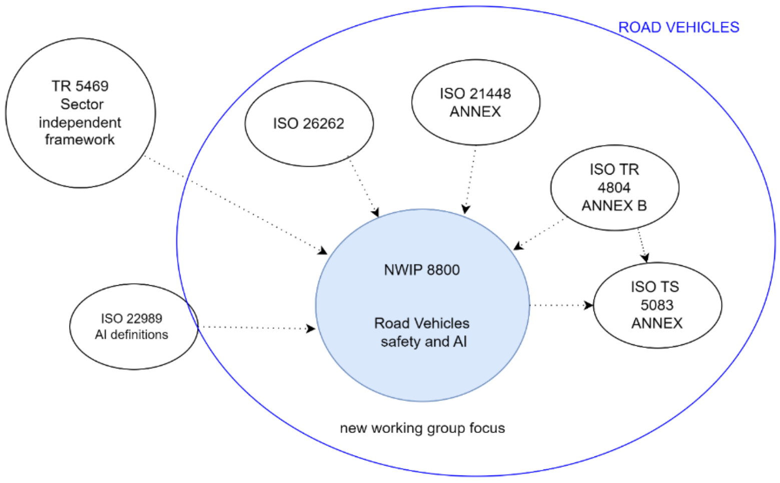


Figure 2: ISO 8800 relation to other standards [15]

- **ISO/TR 4804 – Road Vehicles- Safety and cybersecurity for automated driving systems- Design, verification and validation**

This technical report provides a summary of widely known safety by design, verification and validation (V&V) methods and recommendations of automated driving systems. The purpose of this summary is to provide an overview about the general steps for developing, verifying and validating automated driving systems, focusing on safety and cybersecurity, and also to propose guidelines and a framework to be followed during the lifecycle of the project. This document also aims to present guidance to deal with the risks introduced by automated driving systems.

This standard includes an Annex B called: "Using deep neural networks to implement safety-related elements for automated driving systems". This annex aims to provide an overview of the challenges for achieving and assuring the safety of DNNs in automated driving systems. This chapter proposes potential solutions and architecture principles that can be used as guidance for the development of supervised deep learning.

In the future, this standard will be replaced by ISO/TR 5083 [16] (next subsection).

- **ISO/TR 5083 - Road vehicles — Safety for automated driving systems — Design, verification and validation**

At the time of writing this technical report is under development and there is no public information, but as it will replace ISO/TR 4804 [13], similar topics will be addressed: guidance for developing and validating an automated safety system for road vehicles, taking into account both safety and cybersecurity.

- **IEEE 2846 – Road Vehicles - Assumptions in Safety-Related Models for Automated Driving Systems**

This standard [17] applies to road vehicles. The purpose is to provide an open, transparent, and technology-neutral standard that offers useful guidance for evaluating the performance of automated driving systems.

It defines a minimum set of assumptions for a set of scenarios regarding reasonably foreseeable behaviours of other road users. This minimum set of scenarios shall be considered in the development and testing phase of safety-related models for automated driving systems (ADS).

This standard defines a list of common attributes, models and methods that help verify whether a safety-related model takes the minimum set of assumptions into consideration.

An informative annex provides several examples to show how the proposed minimum set of assumptions could be employed in ADS development.

This standard, specifically, covers:

- Approaches to identify the applicable defined scenario(s)
- Approaches for determining the applicability of assumptions for the given scenario(s) and for updating these assumptions across the temporal evolution of a scenario.
- Approaches to qualify and validate assumptions considering different kinds of performance targets of interest.

- **VDE-AR-E2842-61 – Generic- Development and trustworthiness of autonomous / cognitive systems**

The VDE-AR-E 2842-61, 'Development and trustworthiness of autonomous/cognitive systems' is German application rule that defines a general framework for developing trustworthy solutions and autonomous/cognitive systems.

Trustworthiness is considered a generic concept mandatory to guarantee functional safety, security, privacy, usability, reliability, and intended functionality (among others). This rule presents a reference life cycle with the logical flow to the involved activities, taking the safety life cycle of ISO 26262 as a reference, analogous to current functional safety standards.

This standard tries to cope with the 'uncertainty' related to artificial intelligence. It defines the Uncertainty Confidence Indicator (UCI) for dealing with these uncertainty-related failures and different UCI levels according to the safety required by the application. This application rule proposes demonstrating the achievement of the specific UCI-level requirements in an assurance case. Nevertheless, VDE-AR-E 2842-61 does not specify how to deal with these uncertainty-related failures or how to define this assurance case.

- **ANSI/UL4600 – Fully autonomous vehicles - Safety Standard for Autonomous Vehicles**

ANSI/UL 4600 is a safety standard focusing on the safety for the evaluation of autonomous products. It is the first standard addressing fully autonomous vehicles such as self-driving cars along with applications in mining, agriculture, maintenance, and other vehicles including lightweight unmanned aerial vehicles.

This standard aims to cover the ability of autonomous products to perform safely and as intended with no human interaction. It also addresses the reliability of the hardware and software needed for machine learning, sensing of the operating environment and other safety aspects of autonomous operation.

The standard's scope includes risk analysis and safety-relevant aspects of design process, testing, tool qualification, autonomy validation, data integrity and human-machine interaction for non-drivers. ANSI/UL 4600 defines Safety Performance Indicators (SPI) as operational metrics for the verification and validation of autonomous systems.

- **CoDANN I and II – Avionics- Concepts of Design Assurance for Neural Networks**

CoDANN I and II ([18]and [19]) reports are the result of the joint effort between the European Union Aviation Safety Agency (EASA) and Daedalean AG between July 2020 and May 2021.

The main goal of this project was to analyse the usage in safety-critical applications of systems employing machine learning/neural networks. Focusing in particular on the challenges with respect to trustworthiness, such as the ability to provide performance guarantees, as well as the applicability of existing functional safety guidance such as DO-178C.

An important outcome of this report is the W-shaped development process. It adapts the classical V- shaped cycle to machine learning applications. It also provides an outline of the essential steps for learning assurance and their connection with traditional development assurance processes.

A follow-up to this first report was made public in 2021. The goals of this second report were:

- Investigate topics left out in first report
- Mature the concept of learning assurance
- Investigate remaining trustworthy AI building blocks

- **EASA Concept Paper: First guidance for Level 1 & 2 machine learning applications**

This document aims at guiding applicants when introducing AI/ML technologies into safety-related systems or in applications related to any domain covered by the EASA Regulation.

This document [20] provides a first set of usable objectives and it only covers an initial set of AI/ML techniques. Therefore, nowadays it does not constitute a definitive or detailed guidance, but it will continue evolving according to a defined roadmap.

It has dedicated sections to explainability and trustworthiness.

- **ISO/IEC 22989 - Information technology - Artificial intelligence — Artificial intelligence concepts and terminology**

This document [21] establishes standardized terminology and describes concepts in the field of AI. The standardized concepts and terminology are needed to help AI technology to be better understood and used by a broader set of stakeholders.

It also allows the comparison and classification of different solutions in terms of resilience, reliability, accuracy, safety, trustworthiness, security and privacy. This classification and comparison help stakeholders to better select the appropriate solutions for their applications.

- **IEC TS 6254 - Information technology — Artificial intelligence — Objectives and approaches for explainability of ML models and AI systems**

When AI is used for taking decisions in a system it is important that people understand how those decisions are made. But providing useful and clear explanations of the behaviour of an AI system is a difficult task. IEC TS 6254 focuses on explainability, which is the capacity of explaining the processes undertaken by the machine learning model and its output in a way that it is understandable for a human being.

It presents objectives, approaches, and methods to reach explainability of Machine Learning (ML) models and AI systems, outputs and results. It also provides guidelines on the applicability and properties of the approaches and methods for improving explainability throughout the AI system's life cycle, as defined in ISO/IEC 22989 [21] At the time of writing, this standard [22] is still at a development phase and information is based on early drafts.

- **ISO/IEC DIS 5338 - Information technology — Artificial intelligence — AI system life cycle processes**

At the time of writing, this standard [23] is still at a development phase and there are no details or information from the main concepts and principles that will be presented on it.

- **Guidance on the Assurance of Machine Learning in Autonomous Systems (AMLAS)**

Even if it is not a standard, this document [24] introduces a methodology for the assurance of Machine Learning for use in autonomous systems. It comprises a set of safety cases and process that allow integrating safety assurance in ML development components as well as the generation of explicit evidence justifying that the components have the acceptable safety level when integrated into autonomous system applications.

It covers the following ML lifecycle stages: ML safety assurance scoping, safety requirements elicitation, data management, model learning, model verification and model deployment.

In AMLAS, safety considerations are only meaningful once scoped within the wider system and operational context.

3.1.3. Standards/guidelines for HPEC platforms

On the platform side, functional safety standards such as those presented in Section 3.1.1, advocate for simple, predictable and proven-in-use solutions. Functional safety standards require the verification of the predictability of behaviour, including properties such as performance, resources, response time and worst-case execution time (IEC 61508-3 7.9.2.14). They recommend determining the use of resources by each process and the distribution of demands under average and worst-case conditions (IEC-61508-7 C.5.20/C.5.22). This may become particularly challenging in modern and emerging high-performance embedded platforms as those needed to run AI based solutions, due to their complex parallel architectures, shared resources, lack of in-service experience and detailed public information...In mixed-criticality systems, standards recommend solutions based on deterministic scheduling methods supported by an upper estimation of execution time and program sequence and timing supervision units such as watchdogs. ISO 26262-11 section dedicated to multicores, warns about the fact that multicores are subject to timing faults and it highlights the importance of independence of execution by dedicated analyses and countermeasures such as, the specification of timing constraints and detection of timing requirement violations, doing an upper estimation of resources, evaluating the influence of hardware and software interactions and evaluating timing and execution failure modes.

In order to bridge this gap between functional safety standards and the increasing complexity of platforms required to achieve the performance needs of AI applications, in this section we summarize existing and emerging guidelines focused on multi-core processors.

- **CAST-32A – Avionics - Certification Authorities Software Team (CAST) position paper**

Based on the need of the aerospace domain to adopt multicore processors, the Certification Authorities Software Team (CAST) published a position paper identifying the main objectives that shall be met by a safety critical airborne system executed on a multicore platform in terms of safety, performance, and integrity. Since its release on 2016, further efforts have been devoted to officialising the outcomes of this position paper on an official guidance by the FAA and EASA, which is collected in AMC-20-193 presented in next subsection.

- **AMC-20-193 – Avionics - General Acceptable Means of Compliance (AMC) for Airworthiness of Products, Parts and Appliances. Use of multi-core processors**

The goal of this AMC produced by EASA, is to identify items that might have an impact in the safety, integrity and performance of airborne system software executed on a multicore processor. It provides guidelines for DO-178 [25] aerospace projects developed in multi-core platforms.

It recommends the practices to be considered when working with multicore processors, including considerations for dynamic allocation and multicore interference mitigation. AMC applies to systems and equipment that contain two or more cores activated and contain software application or hardware item with safety implications. Even if the AMC is specific for the aerospace domain, most of the approaches and methods described on it could be applied to other domains handling the development and certification of safety-critical systems on multicore platforms.

AMC-20-193 supplements the guidance in CAST-32A position paper.

3.1.4. Summary of standards of interest for the project

This section summarizes the properties that make previously introduced standards and guidelines interesting for SAFEXPLAIN. Table 6 categorize them based on their relation to functional safety, the guidance they provide for AI based developments, and the extend at which the topics of explainability and high-performance platforms are covered. This is represented by the following symbols: ‘-’ (none or very little), ‘+’ (low), ‘++’ (medium) and ‘+++’ (high). In addition, Table 6 also describes if each document is already published, if there is an available draft or if it is still under development (ongoing) and it specifies the domain of application.

Based on this, the initial phases of SAFEXPLAIN will mainly focus on the following standards and documents (marked in blue colour in Table 6) and will closely monitor the standardization evolution on the presented topics:

- For functional safety, IEC 61508 is taken as reference standard. As a generic industrial standard for functional safety, it allows identifying and stablishing cross-domain safety principles. However, consistency with domain specific functional safety standards for automotive (ISO 26262), railway (EN 5012x) and space (ECSS standard set) will continuously be checked.
- Similarly, for AI and autonomous systems, ISO/IEC 5469 [26] is envisioned to cover functional safety and artificial intelligence in a domain-independent way, and it will probably share many commonalities with the ongoing automotive ISO/PAS 8800 [27]. In addition, ISO/PAS 21448 [12] stablishes the procedures and requirements to guarantee Safety of the Intended Functionality (SOTIF), and even if it is targeted to the automotive domain, the project will identify the common principles applicable to autonomous systems in other domains. In addition, ISO/PAS 21448 complements ISO/IEC 5469 and ISO/PAS 8800 with a specific Annex for ML. Based on the main safety principles derived from these standards, later stages of the project may consider others to further extend specific topics (e.g., ISO/TR 4804 [13] that will be replaced by ISO/TR 5083 for verification and validation methods, and VDE-AR-E2842-61 [28] and UL 4600 [29] for uncertainty management).
  Other standards, such as ISO/IEC DIS 5338 [23] and ISO/IEC 22989 [14] for IT, will also be considered to a lesser extend for terminology and AI system life cycle processes, as recommended by ISO/IEC 5469.

- To deal with the increasing hardware platform complexity, mainly AM(C) 20-193 is [30] considered, as it is the evolution of CAST-32A [31].

It should be noted that the spectrum of available and emerging standards is very wide, with many other additional standards for other domains where autonomous systems can be applied, such as, agriculture, machinery, or robotics. At this stage of the project, these standards are not considered as they do not fit within the application domain of the case studies.

Table 6: Summary and classification of standards

Standards / guideline	Status	Functional safety	AI / ML / DL	Explainability	HPEC platforms	Domain
FUNCTIONAL SAFETY						
IEC 61508	PUBLISHED	+++	-	-	-	**Generic**
ISO 26262	PUBLISHED	+++	-	-	-	**Automotive**
EN 5012x	PUBLISHED	+++	-	-	-	**Railway**
ECSS-Q-ST-40C	PUBLISHED	+++	-	-	-	**Space**
ECSS-Q-HB-80-03A	PUBLISHED	+++	-	-	-	**Space**
ECSS-Q-ST-30C	PUBLISHED	+++	-	-	-	**Space**
DO-178C	PUBLISHED	+++	-	-	-	Avionics
AI AND AUTONOMOUS SYSTEMS						
ISO/IEC TR 5469	ONGOING	+++	+++	+	-	**Generic**
ISO/PAS 21448	PUBLISHED	++	++	-	-	**Automotive**
ISO/IEC TR 24029	PUBLISHED	-	++	+	-	**Generic**
ISO/PAS 8800	ONGOING	+++	+++	-	-	**Automotive**
ISO/TR 4804	PUBLISHED	++	++	-	-	**Automotive**
ISO/TR 5083	ONGOING	++	++	-	-	**Automotive**
VDE-AR-E2842-61	PUBLISHED	+	+	-	-	**Generic**
ANSI/UL 4600	PUBLISHED	+	+	-	-	**Automotive**
CoDANN I and II	PUBLISHED	++	+++	+	-	Avionics
EASA Concept Paper	PUBLISHED	-	+++	++	-	Avionics
ISO/IEC 22989	PUBLISHED	-	+++	-	-	IT
ISO/IEC TS 6254	ONGOING	-	+++	+++	-	IT
ISO/IEC DIS 5338	DRAFT	-	+++	-	-	IT
IEEE 2846	PUBLISHED	+	-	-	-	**Automotive**
AMLAS	PUBLISHED	-	+++	-	-	**Generic**
HPEC / MULTICORE PLATFORMS						
CAST-32A	PUBLISHED	-	-	-	+++	Avionics
AM(C) 20-193	PUBLISHED	-	-	-	+++	Avionics

3.2. Baseline safety principles

The project will adopt an incremental strategy based on safety patterns where DL is introduced in the system with different roles and relevance. The safety implications and requirements of the DL component will determine the safety principles that shall be applied to it. In this sense, ISO / IEC TR 5469 classifies the application of the AI technology on the safety system based on 6 usage levels:

- Level A1 applies to safety-relevant systems where AI technology is used and it is possible to make automated decision of the system function using AI technology.
- Level A2 applies to safety-relevant systems where AI technology is used but it is not possible to make automated decision of the system function using AI technology (e.g., AI technology is present in the system for diagnostics).
- Level B1 applies to safety-relevant systems where AI technology is only used in the development phase (e.g., an offline support tool) and automated decision making of the function developed using AI is possible. This usage level is out of scope of SAFEXPLAIN.
- Level B2 applies to safety-relevant systems where AI technology is only used in the development phase (e.g., an offline support tool) but no automated decision making of the function is possible. This usage level is out of scope of SAFEXPLAIN.
- Level C applies to safety-relevant systems where AI technology is not part of a safety function but can have an impact on it.
- Level D applies to safety-relevant systems where AI technology is not part of a safety function and does not have an impact on it due to sufficient segregation and behaviour control.

Depending on the DL-usage level and the complexity of the ML technology, ISO / IEC 5469 provides different recommendations and requirements. Next subsections summarize the baseline safety principles in terms of safety lifecycle and architectural design.

3.2.1. Safety lifecycle

Functional safety standards, such as, IEC 61508, ISO 26262 or EN 5012x, define well known procedures and technical requirements for the management of functional safety across the complete product lifecycle. While many of these considerations can be kept when developing ML-based systems, some aspects shall be extended to cover specific aspects of ML, like data management, training process or new hazards involved by the nature of AI algorithms. As summarized in previous Subsection 3.1, emerging standards and initiatives are working on this direction. This subsection aims to relate these novel approaches with the traditional functional safety lifecycle.

Next Figure 3 depicts the high-level relation of the FUSA lifecycle with the AI lifecycle approach proposed by AMLAS. As it can be seen, the FUSA lifecycle follows a V-model approach, which can be decomposed into hardware and software V-models. When incorporating ML components in the system architecture design, the traditional software V-model shall be accommodated to ML particularities. Following AMLAS, we introduce the concepts of “data management”, “model learning” and “model verification” into the SW module realization phases (design, implementation and testing). However, the inclusion of ML components has further implications in all phases of the lifecycle, starting from the hazard and risk analysis, requirements specification, up to model deployment and validation. Therefore, additional phases for “ML safety assurance scoping”, “ML requirements” and “ML deployment” are contemplated.

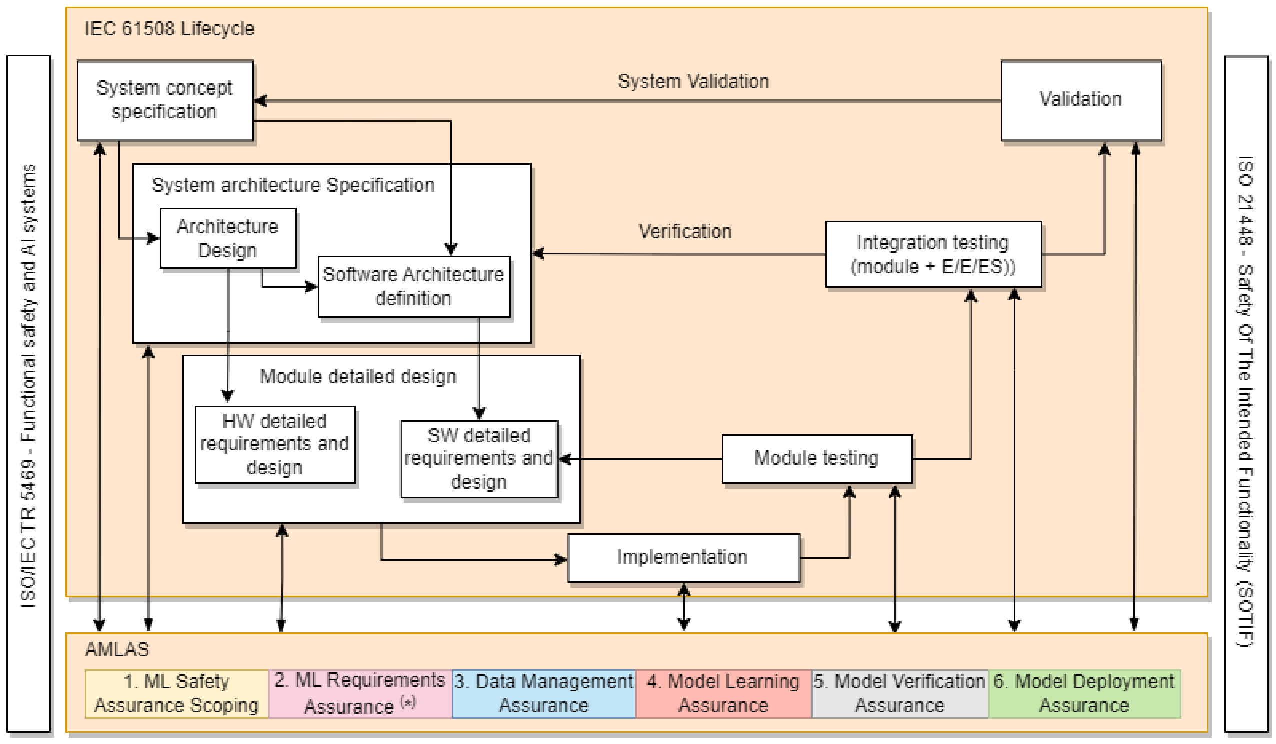

IEC 61508 Lifecycle
System concept specification
System Validation
Validation
System architecture Specification
Architecture Design
Software Architecture definition
Verification
Integration testing (module + E/E/ES))
Module detailed design
HW detailed requirements and design
SW detailed requirements and design
Module testing
Implementation
AMLAS
1. ML Safety Assurance Scoping
2. ML Requirements Assurance (*)
3. Data Management Assurance
4. Model Learning Assurance
5. Model Verification Assurance
6. Model Deployment Assurance
ISO/IEC TR 5469 - Functional safety and AI systems
ISO 21448 - Safety Of The Intended Functionality (SOTIF)

Figure 3: Relation between FUSA lifecycle (IEC 61508) and AI lifecycle (AMLAS)

The process proposed in AMLAS is aligned with the requirements and recommendations provided by the ISO/IEC TR 5469 standard, which in many cases refers to IEC 61508 techniques even for ML systems development. In Table 7, we summarize main ISO/IEC TR 5469 requirements and recommendations together with AMLAS guidance of each of the AI lifecycle step.

Table 7: ISO / IEC TR 5469 and AMLAS requirements and recommendations for each AI lifecycle step

	ISO / IEC TR 5469	**AMLAS**
ML Safety Assurance Scoping	The current draft indicates that: - Requirements or properties can be based on existing standards although other may need to be newly defined. - Hazard and risk analysis phase can be based on IEC 61508 or other functional safety standards, with some modifications to address the black-box nature of ML.	Definition of the system safety requirements, description of the system, ML component (including its interfaces) and operating environment of system. Hazard identification and risk analysis. Safety requirement allocation from system to ML component.

	ISO / IEC TR 5469	AMLAS
ML Requirements Assurance	In its current draft, it does not define specific requirements for ML requirement specification. Instead, it makes the following analysis: - Common FUSA practices can be applied (e.g., IEC 61508-3 techniques for software requirements specification). - In some cases it is difficult to define safety requirements for AI algorithm and refers to IEC TS 62988-1 and IEC 61496 as an example of the definition of a person detection function.	Specification of the ML safety requirements from the allocated system safety requirements. Refinement the ML safety requirements into performance and robustness requirements. Definition of the assumptions about the system or operating environment. ML safety requirements verification.
Data Management Assurance	Training data: - has to provide complete representativeness of the input domain. - has to reflect the distribution of the application context. Identification of the sources of data drift (seasonal change, changes in the process that can induce this data drift, unforeseen input by operators, different lighting conditions in training data than operational data …) and establish correction of the model (re-training, the estimation and inclusion of correction factors, supervision correction...). Test data: - Shall be representative of all safety-relevant scenarios identified during HARA. - Shall cover variations of situations involving safety risks. - Shall be diverse enough and sufficient to verify the proper training (right training outcome) for those safety-relevant scenarios from HARA and the mentioned variations. - Shall ensure the stable outcome of testing for those safety-relevant scenarios from HARA and the mentioned variations. Clearly specify sets of data attributes that lead to each of the safety risks identified in HARA.	Data requirements shall specify the characteristics that the collected data have to meet, including relevance, completeness, accuracy and balance: - Relevance requirements: shall specify the extent to which the data match the intended operating domain into which the model is to be deployed. - Completeness requirements: shall specify the extent to which the development data must be complete with respect to a set of measurable dimensions of the operating domain. - Accuracy requirements for the data shall be specified. - Balance requirements shall specify the required distribution of samples in the data sets. A data requirement justification report shall be provided that the specified ML data requirements are sufficient to ensure it is possible to develop a machine learning model that satisfies the ML safety requirements of previous phase. Training and verification data should be collected by different people/teams. Decisions made when collecting, processing and augmenting the data should be recorded to explain how the data sets meet the data requirements. A data generation log shall be kept, which details the decisions made in each sub-process to obtain data with the desired features.

	ISO / IEC TR 5469	AMLAS
Model Learning Assurance	Ensure the independence between test and training data and therefore, independence between the teams collecting the data and the teams performing the tests. Ensure that data are free of malicious modifications or alterations (ensure the credibility of data source and data collection processes).	Generate data sets in accordance with data requirements for the development and verification stages (training, verification, validation). Analyse the datasets to verify whether they can meet the data requirements and it should be explicitly documented. This data set should consider accuracy, relevance, completeness and balance of the data sets. When existing data sets are re-used, additional validation tasks may be required to ensure that the labels are sufficient for the context into which the model is to be deployed.
	Achieve an appropriate level of transparency: should not be as low as poses risks in terms of fairness, security, and accountability, neither as high than lead to confusion due to information overload. A trade-off between explainability and performance of the system, reaching an overall performance in terms of the quality of the decision achieving the highest level of explainability. The information about the AI system's decision-making should be clear, coherent, accurate, complete and understandable by experts and end users. A methodical and formally documented evaluation of model interpretability shall be done, subject to evaluation of the consequences on functional safety risks. Incorporate forms of concept drift detection that differentiate between drift and noise present in the system, allowing to adapt the model over time. Analyse the vulnerability factors associated with AI technology (special emphasis in random HW failures). Guarantee portability between training and inference platforms (i.e.,	Record the decision of the most appropriate ML model for the problem and for satisfying safety requirements, and the expert knowledge or previous experience that led to its election. The process followed in generating the ML model has to be documented, justifying the key decisions (including the choice of the development tools) and those choices that impact on the performance and robustness of the model. The results from evaluating the ML model with the internal test data have to be documented, providing evidence of satisfying the ML safety requirements.

	ISO / IEC TR 5469	AMLAS
	translational errors due to memory incompatibilities of data management). Fault detection during training (e.g., ground truth verification, cross checking ...).	
Model Verification Assurance	Detect and mitigate training errors during the training phase. Safety risk mitigation techniques shall clearly address each safety risk identified during HARA and its role in maintaining functional safety. Approaches focused on guaranteeing interpretability or explainability of the model are desirable (how an AI system is constructed, analysis of the model to gain a partial "explanation" of the output behaviour…).	For each ML safety requirement at least one verification activity shall be undertaken. The verification activities: - should be sufficiently independent of the development activities and include the measures taken to verify the ML model. - shall be comprehensive. - shall clearly demonstrate coverage with respect to variability and combinations relevant to the ML safety requirements. The results of the ML safety requirements verification have to be recorded. The verification strategy should include the range of tests undertaken and the rationale for performing each test with bounds and test parameters where appropriate. In addition, the approaches taken to manage verification data in such a way as to ensure that data leakage did not occur should be documented.
Model Deployment Assurance	Once an AI system is approved and in operation, its own incident statistics can be used to provide ongoing evidence of safety performance. Most IEC 61508-3 techniques can be applied for safe model deployment, including: - Fault detection during inference. - Diverse monitor and diverse redundancy. - Cyclic behaviour, with guaranteed maximum cycle time. In addition to the previous techniques proposed by IEC 61508-3, IEC 5469 defines a set of control and mitigation measures related to architectural considerations (supervision function, redundancy, ensemble concept and diversity...).	Measures shall be put in place to monitor and check validity throughout the operation of the system of the key system and environmental assumptions. Mechanisms shall be put in place to mitigate the risk posed if any of the assumptions are violated. The system shall monitor the outputs of the ML model during operation, as well as the internal states of the model, together with erroneous inputs to identify when erroneous behaviour occurs. erroneous input, outputs, and model states, shall be documented in the erroneous behaviour log. When integrating the model into the system the suitability of the target hardware platform shall be considered The system in which the ML model is deployed shall be designed such that the

	ISO / IEC TR 5469	AMLAS
		system maintains an acceptable level of safety even in the face of the predicted erroneous outputs that the model may provide.

3.2.2. Safe Inference Platform

For the inference of the ML-based system, traditional functional safety approaches shall be followed for the mitigation and control of possible errors. However, the intrinsic complexity of the HW and SW that build these systems challenge the efficiency and coverage that can be attained with existing techniques. In addition, with ML systems, the concept of unintended behaviour (or model insufficiencies) shall be handled too. ISO/IEC TR 5469 classifies the elements of the ML model as:

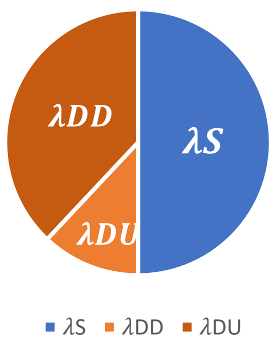
- Application independent technology elements: executable code, low level libraries, set of calculations… In general, safety techniques of traditional functional safety standards could be applied.
- Application dependent technology elements: application graph, deep learning model… These elements depend heavily on data and are likely not covered by existing techniques in FUSA standards.

This subsection explains basic concepts and common architectural design patterns on FUSA standards and provides an initial overview on how they relate to ML-based solutions together with the recommendations of ISO/IEC TR 5469 on the platform.

3.2.2.1. Architectural safety patterns from traditional FUSA (IEC 61508)

Functional safety standards such as IEC 61508 and those based on it (e.g., ISO 26262 and IEC 5012x) define the average probability of dangerous failure of a safety function according to its SIL. In addition, the highest SIL that can be claimed for a safety function is limited by the following concepts (see Table 8):

- Hardware fault tolerance (HFT): is the ability to perform the safety function in the presence of N faults. An HFT of N means that N+1 faults could cause the loss of the safety function.
- Safe Failure Fraction (SFF): ratio of average safe (λS) + dangerous detected (λDD) failure rates and total average failure rate (i.e., safe (λS) + dangerous (λD), where λD is the sum of dangerous detected (λDD) and dangerous undetected (λDU)):



$$SFF = \frac{\sum \lambda S + \sum \lambda DD}{\sum \lambda S + \sum \lambda DD + \sum \lambda DU} \qquad \text{(eq. 1)}$$

Table 8: Maximum allowable safety integrity level according to IEC-61508-2 for Type B safety related elements

Pattern	HFT	Safe Failure Fraction (SFF)			
		<60%	60% -90%	90% - 99%	≥ 99%
1oo1(D)	0	Not allowed	SIL 1	SIL 2	SIL 3
2oo2(D)	0	Not allowed	SIL 1	SIL 2	SIL 3
1oo2(D)	1	SIL 1	SIL 2	SIL 3	SIL 4
2oo3(D)	1	SIL 1	SIL 2	SIL 3	SIL 4
1oo3(D)	2	SIL 2	SIL 3	SIL 4	SIL 4

According to eq. 1, the SFF can be improved by increasing the portion of safe and dangerous detected failure rates and by correspondingly reducing the dangerous undetected (λDU) ones. This can be achieved through diagnostic mechanisms that provide the required Diagnostic Coverage (DC):

$$DC = \frac{\sum \lambda DD}{\sum \lambda DD + \sum \lambda DU} \qquad \text{(eq. 2)}$$

Next, we summarize different generic architecture patterns to achieve the required SIL, HFT and SFF requirements.

- **Single channel architecture without diagnostics (1oo1, HFT = 0)**

A safety function is usually performed by more than one subsystem (e.g., sensor, logic, actuator). In single-channel architectures, this is a serial combination of elements as shown in the example of Figure 4. In this case, the maximum SIL of the safety function is determined by the element with the lowest SIL (i.e., SIL 1 in the example of Figure 4) and the probability of dangerous failure of the safety function is the addition of the probabilities of dangerous failure of all elements.

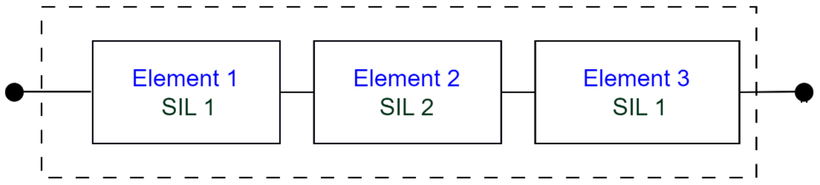


Figure 4: Single channel without diagnostic (1oo1)

In this architecture, any dangerous failure of a subsystem element, causes a failure of the safety function. According to Table 8, with this architecture, the safe failure fraction shall be at least 60% for SIL 1 safety functions and higher for greater SILs. To this end, the safe portion of the failure rate shall be high enough or otherwise diagnostic mechanisms shall be applied as explained in next architecture.

- **Single channel architecture with diagnostics (1oo1D, HFT = 0)**

In order to increase the safe failure fraction of each element, periodic diagnostic mechanisms, executed every Diagnostic Test Interval (DTI), shall detect the error and perform an action (refer to Figure 5 for an example) to reach and maintain the safe state in a period of time that prevents the hazardous event. In this case, undetected dangerous failures of a subsystem element cause the loss of the safety function.

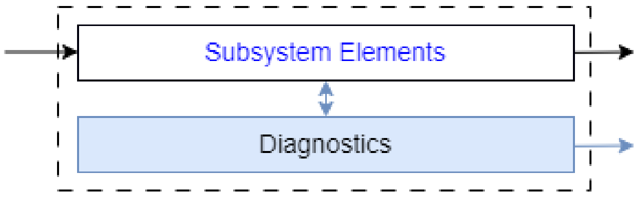


Figure 5: Single channel with diagnostics (1oo1D)

- **Dual-channel architecture (2oo2(D), HFT = 0)**

The 2oo2 is a two-channel architecture that consists of two parallel channels, each performing the safety function. A voter checks the consistency among both channels and in case of discrepancy, it performs an action to reach and maintain the safe state. Therefore, in this architecture, both channels shall be error free in order to keep the execution (i.e., HFT = 0) but higher Diagnostic Coverage than in previous approaches is easier to achieve thanks to the comparison voting. Common cause failures among the two channels shall also be contemplated, as they could result in dangerous undetected system failures. A way to reduce this, is by employing diversity, i.e., by using different solutions and technologies to build the redundant channels.

As in the single channel architecture, the dual channel architecture can be implemented with additional diagnostics (2oo2D), reducing the amount of dangerous undetected faults.

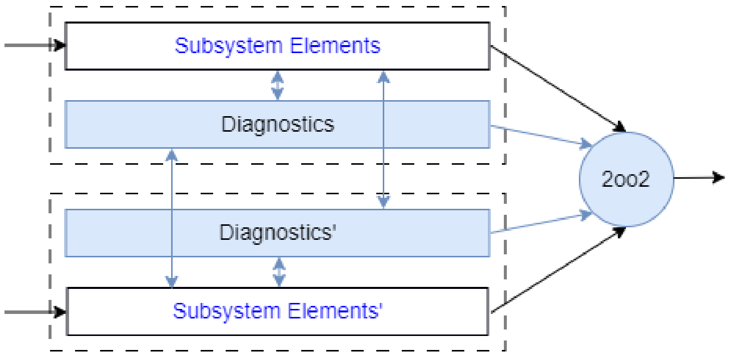


Figure 6: Dual channel with diagnostics (2oo2D)

- **Dual-channel architecture (1oo2(D), HFT = 1)**

This dual-channel architecture is similar to previous, but instead of comparing the outputs, each channel has the ability of performing the safety function independently (e.g., each opening a safety relay and both connected in series as shown in the example of Figure 7). In this case, a single fault does not lead to the loss of the safety function, as the second channel is still able to run it (HFT = 1). Simultaneous dangerous failures on both channels could lead to loss the safety function, therefore common cause failures shall be considered.

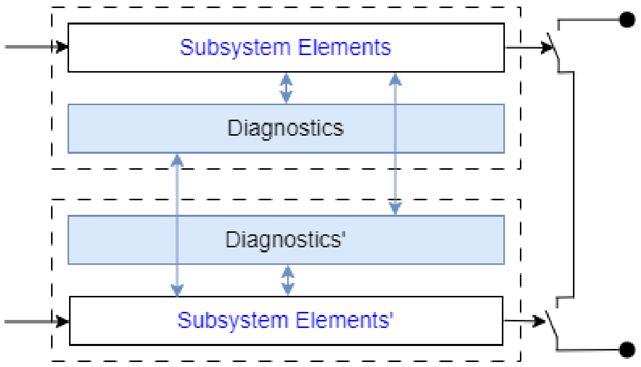


Figure 7: Dual channel with diagnostics (1oo2D)

- **Triple Module Redundancy (TMR) with majority voter (2oo3, HFT = 1)**

Previous concepts can be extended to an increasing number of channels, such as triple-channel architectures. A common approach is to use a majority voter to select the correct output among the three channels. This means that a single fault can be detected and tolerated (HFT = 1) but two faults in combination can lead to the loss of the safety function. As in previous cases, this architecture is commonly complemented with diagnostics mechanisms in each channel and with cross-monitoring among channels.

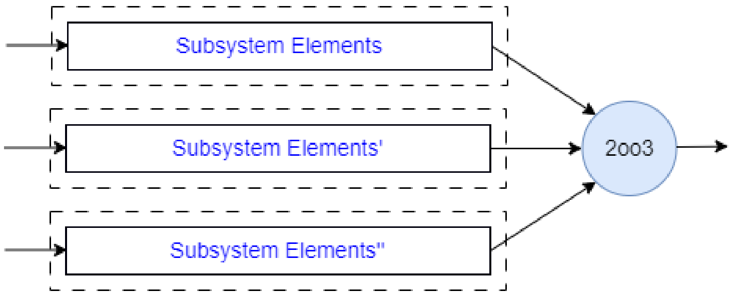


Figure 8: Triple Module Redundancy (TMR) with majority voter (2oo3)

- **Triple Module Redundancy (TMR) (1oo3, HFT = 2)**

Instead of applying the majority voter, each channel could also independently perform the safety function, analogously to the example explained in the 1oo2 dual channel architecture. In this case, the 3 channels need to fail in order to lose the safety function, and therefore 2 faults are tolerated (HFT = 2). As in previous cases, this architecture is commonly complemented with diagnostics mechanisms in each channel and with cross-monitoring among channels.

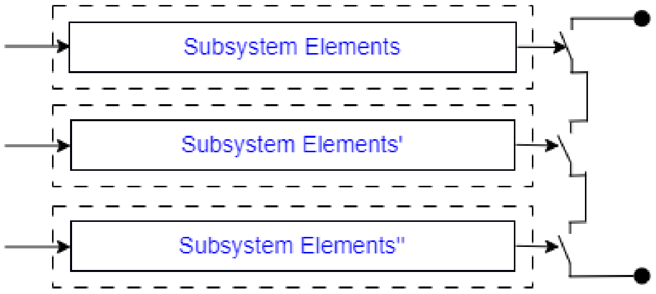


Figure 9: Triple Module Redundancy (TMR) (1oo3)

3.2.2.2. Architectural safety patterns for AI

Same architectural properties described in previous subsection can be applied for safety systems with AI components with some particularities. Following the ISO / IEC TR 5469 approach, architectural safety patterns for AI could include 3 levels of complementary mechanisms depicted in Figure 10 and explained below: redundancy and diversity, monitors / detection mechanisms, and supervisors.

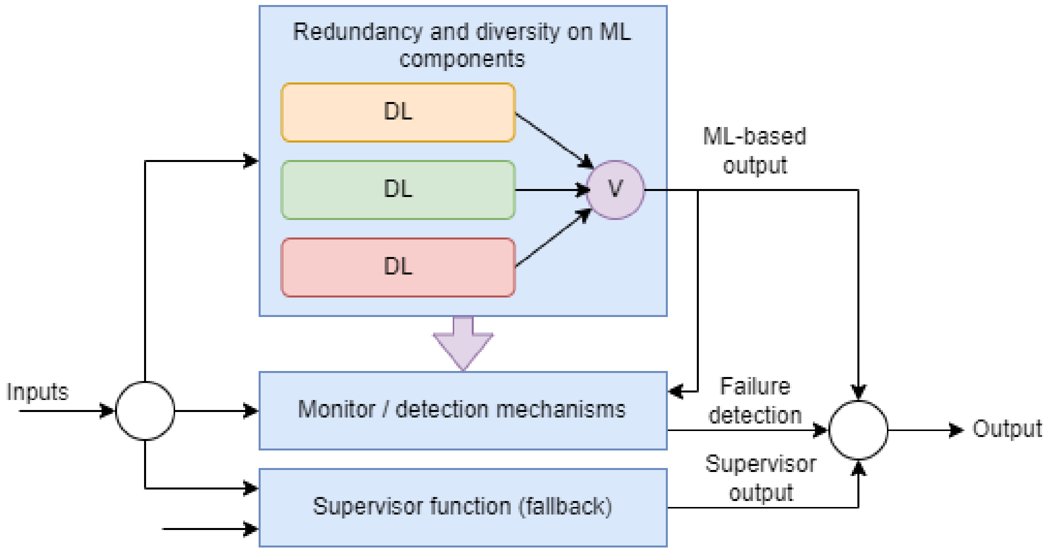

Redundancy and diversity on ML components
DL
DL
DL
V
ML-based output
Inputs
Monitor / detection mechanisms
Failure detection
Output
Supervisor output
Supervisor function (fallback)

Figure 10: Complementary architectural patterns for safe AI [32]

The selection of techniques and the design of the specific architecture will depend on the safety requirements of Table 8 such as SIL, HFT and SFF. In addition, the DL-usage level introduced in Section 3.2, will determine the SIL, HFT and SFF requirements allocated to the ML component.

1) <u>Redundancy and Diversity</u>

As shown in previous Table 8, in order to meet the IEC 61508 requirements for SIL 3, some form of redundancy is required, unless it is proven that the SFF is higher than 99%, which can be very challenging in complex systems with AI. Following previously described architectural patterns, several DL-models can be used to build a redundant architecture. In addition, for higher robustness and effectiveness, redundancy can be combined with diversity. This diversity can be introduced at different points as proposed by the authors of [33].

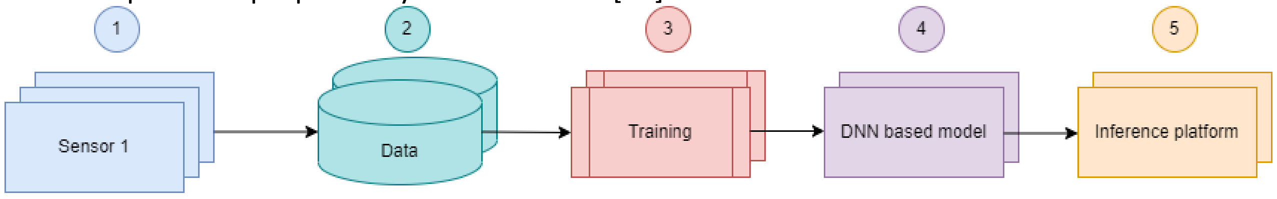


Figure 11: Diversity approaches [33]

1. Inputs: different sensors set in distinct locations/positions can be used as input sources in order to capture the same object in varying angles, with different perspective, divergent physical principles of measurement, and sensor-specific characteristics.
2. Training data: models can be trained by different input data sets, different labelling rules, which will result in different models.
3. Training: different technologies, processes and people can be employed to train the several redundant models. For example, changing the initialization point of the weights, either with the same or different data for training, will potentially result in diverse models.

4. DL model technology: with diversity of AI technology itself, selected algorithm, number of layers, or its configuration.
5. Inference platform: the redundant DL models can be deployed into different hardware platforms and/or with different low-level software libraries.

These diversity approaches shall be combined to sufficiently cover potential faults on the hardware and software (mainly (4) and (5)) as well as model insufficiencies caused by an incorrect or insufficient training process (1), (2), (3), (4).

2) <u>Monitors / detection mechanisms</u>

In the same way as for traditional FUSA approaches of Subsection 3.2.2.1, redundancy and diversity techniques can be complemented with diagnostics. The usage of a monitor and diagnostic mechanisms can detect when an AI technology is producing potentially unsafe actions, either due to functional insufficiencies or due to a fault. Therefore, traditional functional safety mechanisms for fault detection shall be complemented with new mechanisms that detect functional insufficiencies.

The monitor or the detection mechanisms oversee the behaviour of the ML component(s) with reference to safety. According to ISO/IEC TR 5469, such mechanisms could include:

- Plausibilization methods that check the model output for consistency (for example checking impossible positions, objects detected, sizes).
- Input observation methods that guarantee that the input is statistically close to the training dataset.
- Attention mechanisms such as heat maps.
- Detection mechanisms for abnormal neural behaviour.
- Safety envelope concept: a typical example of this method is the doer/checker approach where a safe and unsafe region is specified. The primary channel implements AI functionality while the other channel supervises the primary channel. These two channels are constantly cross checking. Once the output of the doer enters the unsafe region, the checker function implements a pre-defined reaction (e.g., safe state).

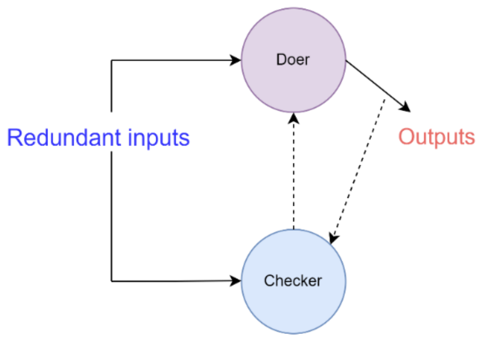


Figure 12: Doer-Checker diagram

With this approach, the checker subsystem could rely on traditional non-AI software and implement the safety functions while the doer is in charge of the untrusted functionality. Therefore, we could have a Doer at low safety integrity level while the Checker must be at higher SIL. According to ISO/IEC 5469, this solution can be considered either usage level C or usage level D.

In all this cases, the monitor must be driven by an independent power supply and can be developed using either non-AI techniques or using AI techniques.

3) Use of a (non-AI) supervisor function

A supervisor consists of a simpler safe controller based on classical deterministic algorithms and with limited capacities. This controller operates independently from the ML-based system and shall be able to reach a safe state. The supervisor verifies the output of the ML-subsystem corresponding to a given input.

Thanks to the presence of a supervisor, in case of disagreement in the outputs generated by the DL based software or any kind of failure/malfunction of DL-based components detected by the supervisor or when the outputs are judged not reliable, the safe state can be reached.

Acronyms and Abbreviations

AMC	Acceptable Means of Compliance
AI	Artificial Intelligence
AMLAS	Assurance of Machine Learning in Autonomous Systems
ADS	Automated Driving Systems
ASIL	Automotive Safety Integrity Level
CAST	Certification Authorities Software Team
DL	Deep Learning
DNN	Deep Neuronal Network
DAL	Design Assurance Level
DC	Diagnostic Coverage
DTI	Diagnostic Test Interval
E/E/PE	Electrical/Electronic/Programmable Electronic
ECU	Electronic Control Unit
ESA	European Space Agency
EASA	European Union Aviation Safety Agency
FAA	Federal aviation Administration
FUSA	Functional Safety
GPU	Graphics Processing Unit
HW	Hardware
HFT	Hardware fault tolerance
HPEC	High Performance and Embedded Computing.
HPC	High Performance Computing
IT	Information Technology
KPI	Key Performance Indicator
ML	Machine learning
MS	Milestome
NR	Not Recommend
PST	Process Safety Time
QoS	Quality of Service
SFF	Safe Failure Fraction
SIL	Safety Integrity Level
SOTIF	Safety of the intended functionality
SPI	Safety Performance Indicators
SoA	Service Oriented Architecture
SW	Software

SDK	Software Development Kit
TMR	Triple Module Redundancy
UCI	Uncertainty Confidence Indicator
VAT	Value Added Tax
V&V	Verification & Validation
WP	Work Package

References

[1] International Electrotechnical Commission, IEC 61508 Functional safety of Electrical/Electronic/Programmable Electronic safety-related systems (Second edition), Geneva, 2010.

[2] International Organization for Standardization, ISO 26262 - Road vehicles — Functional safety, 2018.

[3] UNE/EN 50128:2012 / UNE-EN 50129:2020 - Railway applications - Communication, signalling and processing systems, 2012 / 2020.

[4] International Organization for Standardization, ISO 13849 - Safety of machinery - Safety-related parts of control systems, Geneva, 2015.

[5] International Organization for Standardization, ISO 10218 Robots and robotic devices — Safety requirements for industrial robots, Geneva, 2011.

[6] International Organization for Standardization, ISO 18497: Agricultural machinery and tractors — Safety of highly automated agricultural machines — Principles for design, Geneva, 2018.

[7] European Commitee for Electrotechnical Standardization , EN 61511: Functional safety - Safety instrumented systems for the process industry sector, Brussels, 2017.

[8] European Cooperation for Space Standardization , ECSS-Q-ST-30C, 2017.

[9] European Cooperation for Space Standardization , ECSS-Q-ST-40C, 2017.

[10] European Cooperation for Space Standardization, ECSS-Q-ST-80C, 2017.

[11] International Organization for Standardization , ISO/IEC 24029 -Artificial Intelligence-Assessment of the robustness of neural networks, Geneva, 2021.

[12] International Organization for Standardization, ISO 21448: Road vehicles — Safety of the intended functionality, Geneva, 2022.

[13] International Organization for Standardization, ISO/TR 4804: Road vehicles — Safety and cybersecurity for automated driving systems — Design, verification and validation, 2020.

[14] ISO/IEC 22989: Information technology — Artificial intelligence — Artificial intelligence concepts and terminology, Geneva, 2022.

[15] Presentation of ISO/PAS 8800 to UNECE, 2021.

[16] International Organization for Standardization, ISO/TR 5083 - Road vehicles — Safety for automated driving systems — Design, verification and validation, Geneva, Under Development.

[17] Institute of Electrical and Electronics Engineers, IEEE 2846 – Road Vehicles - Assumptions in Safety-Related Models for Automated Driving Systems, 2022.

[18] Concepts of Design Assurance for Neural Networks, CODANN I, 2020.

[19] Concepts of Design Assurance for Neural Networks , CODANN II, 2021.

[20] First guidance for Level 1 & 2 machine learning applications, 2023.

[21] International Organization for Standardization, ISO/IEC 22989 - Information technology - Artificial intelligence — Artificial intelligence concepts and terminology, Geneva, 2018.

[22] International Electrotechnical Commission, IEC TS 6254 - Information technology — Artificial intelligence — Objectives and approaches for explainability of ML models and AI systems, Geneva, Under Development.

[23] ISO/IEC DIS 5338 - Information technology — Artificial intelligence — AI system life cycle processes., Under Development.

[24] Guidance on the Assurance of Machine Learning in Autonomous Systems (AMLAS), 2021.

[25] DO-178: Software Considerations in Airborne Systems and Equipment Certification, 2011.

[26] ISO/IEC 5469 - Artificial intelligence – Functional safety and AI systems, Geneva, Under development.

[27] ISO/AWI PAS 8800 Road Vehicles — Safety and artificial intelligence, Under Development.

[28] Verband der Elektrotechnik, Elektronik und Informationstechnik, VDE-AR-E 2842-61: Development and trustworthiness of autonomous/cognitive systems, Frankfurt am Main, 2021.

[29] Underwriters Laboratories, UL 4600: Fully autonomous vehicles - Safety Standard for Autonomous Vehicles, Northbrook, 2022.

[30] General Acceptable Means of Compliance , General Acceptable Means of Compliance (AMC) for Airworthiness of Products, Parts and Appliances. Use of multi-core processors, 2022.

[31] Certification Authorities Software Team (CAST) position paper, CAST-32A, 2022.

[32] R. Mariani, "Safety and Artificial Intelligence," in *14th TÜV Rheinland Symposium - Functional safety and cybersecurity in industrial applications*, Cologne, Germany, 2022.

[33] Institute of Electrical and Electronics Engineers, IEEE- On Neural Networks Redundancy and Diversity for Their Use in Safety-Critical Systems, 2019.

SC												
ID	Author	WP1	WP2	WP3	WP4	WP5auto	WP5rail	WP5space	Description			
									Subject	Shall/sho	what	when
R2.1	IKR		X						Safety assurance case templates	shall	be provided to define the arguments and identify the evidence that demonstrates that DL subsystem is safe for its purpose	during development
R2.2	IKR		X						Safety assurance case templates	shall	include a definition of processes, methods and lifecycle considerations for DL-software specification, design and implementation based on selected standards	during development
R2.3	IKR		X						Safety assurance case templates	shall	include a definition of processes, methods and lifecycle considerations for DL-software V&V	during development
R2.4	IKR		X						New methods for safety analysis	shall	be provided to consider the quantification of DL-software failure rate in the assessment of the overall system residual risk	during development and operation
R2.5	IKR		X						Different safety architectural design patterns for safe DL deployment (HW and SW techniques)	shall	be defined and explored	during development
R2.6	IKR		X						The architectural design patterns for safe DL deployment (HW and SW techniques)	shall	consider mechanisms to reduce the impact of adversarial attacks	during development and operation
R2.7	IKR		X						The architectural design patterns for safe DL deployment (HW and SW techniques)	shall	be tailored to specific platforms	during development
R2.8	IKR		X						A safety-security concept for a selected case-study	shall	be developed and assessed by an external certification authority	during development
R2.9	IKR		X						New guidelines and/or adaptations to existing and ongoing standards	should	be proposed based on feedback obtained from external certification authorities and domain experts (e.g., ESA)	during development
R2.10	IKR	X							All partners	shall	use as reference the same selected and agreed standards from where a common and useful substrate should be extracted	during development
R23.1	*BSC*		*X*	*X*					A common set of definitions and means of compliance (including evaluation metrics) required from WP3 to WP2 to achieve safety goals	shall	be agreed between WP2 and WP3, support the following evaluation dimensions of DL components (model/data) under test: explainability, traceability, safety, robustness, confidence;	*during development and testing*
R3.1	BSC			X					AI library and the underneath framework on which it builds	shall	provide visibility (i.e. we need to see the source code of the AI library) to understand it and potentially change it	during testing
R3.2	BSC			X	X				FUSA-aware DL library implementations	shall	achieve sufficient performance and adherence to safety standards	during testing
	BSC											
R3.3	RISE			X					The explanations of DL components	shall	support different presentations (text, tabular, graph, visual) upon req. from safety. Low-dimensional human understandable concepts (text representation) shall be extracted, where applicable and feasible	during development
R3.4	RISE			X					The explanations of DL components	shall	support local explanation (explain specific case) and global explanation (explain global behaviour of the DL components)	during development
R3.5	RISE			X					DL components	shall	be designed and implemented in accordance with the safety requirements and constraints determined in WP2	during development
R3.6	RISE			X	X				Reference safe architectures of DL components	shall	be proposed that include the usage of supervisor (safety cage architecture)	during development
R3.7	RISE			X					Datasets used for DL training/test/validation	shall	be evaluated with the proposed metrics in R23.4 and should cover: relevance, completeness, balance and accuracy	during development
R3.8	RISE			X					Approaches to improve dataset coverage	shall	be proposed in accordance with the metrics (R3.7) that include using GAN-based algorithms to generate relevant data points	during development
R3.9	RISE			X					Explainable DL approaches	shall	be proposed for all DL components that are used by 03 UCs to meet the safety requirements	during development
R3.10	RISE			X					Explainable DL components	shall	be prototyped for popular classes (examples: *MLP, CNN, RNN/LSTM, Transformers*)	during testing and operation
R3.11	RISE			X					The DL components	shall	provide explanations in alignment with information requirements from the defined V&V process	during development
R3.12	RISE			X					The DL components	shall	provide traceability in alignment with information requirements from the defined V&V process	during development
R3.13	RISE			X					The DL components	shall	be measurable and characterizable by the evaluation metric categories as in R23.4	during development
R3.14	*BSC*		*X*	*X*					a proof-of-concept for how to specify DNNs as a collection of explainable modules together with data/model specification	shall	be provided by WP3	during testing
R345.1	*EXI*			*X*	*X*	*X*	*X*	*X*	*The Case Studies*	*shall*	*use the standardized APIs provided by the platform software, if available*	*during testing and operation*
	BSC			*X*		*X*	*X*	*X*	*Case studies*	*shall*	*provide information about layers, type of layers, sw components and dataflow, all hyper-parameters that identify a DL model and that can require explainable requirements*	*during testing*
R35.1												
R35.2	*RISE*			*X*	*X*	*X*	*X*	*X*	*The defined evaluation criteria/metrics in R23.4*	*shall*	*be measureble for the proposed DL components (model/data) and approaches that are needed to support 03 UCs*	*during development and testing*
R35.3	*AIKO*			*X*		*X*	*X*	*X*	Explainable DL components	shall	be usable to evaluate the case studies	*during testing and operation*
R4.1	BSC	X	X		X				Platform	should	provide adequate observability channels (e.g Performance Monitoring Counters) on CPU, SoC and interconnect events	during testing
R4.2	BSC	X			X				BSP/Hypervisor/RTOS	should	allow (not prevent) configuration of control registers for QoS and collection of hardware events	during testing
R4.3	BSC	X			X				Platform	should	provide debug support via IDE for Windows and Linux	during testing
R4.4	BSC	X			X				Platform	should	allow collecting end-to-end timing of selected libraries and applications	during testing
R4.5	BSC				X	X	X	X	Platform and case study	shall	shall allow reproducibility of the results by allowing the control of the execution conditions (hardware state)	during testing and operation
R4.6	BSC				X	X	X	X	Platform and case study	shall	shall allow reproducibility of the results by allowing the control of the software state, and input vectors/space	during testing and operation
R4.7	BSC								The monitoring platform	shall	allow collecting a relevant set of metrics identified in WP3, with special focus on those meant to support understandability and explainability.	during testing
R4.8	BSC				X				Library enabling platform-level support for timing interference control/mitigation	shall	be provided	during testing
R4.9	BSC				X				A library enabling collection of timing information on top of the target HW and SW configuration	shall	be provided	during testing
R4.10	BSC				X				A statistical method and tool for timing analysis of DL libraries and use cases	shall	be provided	during testing
R4.11	BSC				X				Hardware configurations to meet FUSA requirements and Safety patterns	shall	be identified	during testing
R4.12	BSC				X				Partially automated support for collecting timing information on the target HW and SW configuration	shall	be developed	during testing
R4.13	BSC				X				Guidelines and practices for statistical predictability approaches, hardware observability & configuration	shall	be developed	N/A
	EXI				X				The platform software	shall	provide standardized facilities to monitor the control flow and manage the execution of multiple applications with mixed criticality	during operation
R4.14												
R4.15	EXI				X				The platform software	shall	provide standardized facilities to manage exchange (transmission and reception) of information between applications	during operation
R4.16	EXI				X				The platform software	shall	provide standardized facilities to manage persistent storage and configuration for applications	during operation
R4.17	EXI				X				The platform	shall	provide data visualization utilities to support the development phase	during testing
R4.18	EXI				X				The platform software	shall	provide standardized facilities to support the execution of integration tests	during testing
R4.19	EXI				X				The platform software	shall	provide facilities to record and replay the case studies	during testing
	BSC	*X*			*X*	*X*	*X*	*X*	*All partners*	*shall*	*select a target platform to run the case studies*	*N/A*
R45.1												
R45.2	*BSC*	*X*			*X*	*X*	*X*	*X*	*All partners*	*shall*	*select a software platform to run the case studies*	*N/A*
R45.3	*BSC*	*X*			*X*	*X*	*X*	*X*	*All partners*	*shall*	*procure the target hardware if needed*	*N/A*
R45.4	*IKR*	*X*			*X*	*X*	*X*	*X*	*The selected platform*	*should*	*support PyTorch and optionally TensorFlow + Keras*	*during use case develop., testing and operation*
R45.5	*IKR*	*X*			*X*	*X*	*X*	*X*	*The selected platform*	*should*	*support NVIDIA Jetpack, TensorRT, python Libraries*	*during use case development, testing and operation*
R45.6	*IKR*	*X*			*X*	*X*	*X*	*X*	*The selected platform*	*should*	*support OpenCV software (needed by frame preprocessing and stereo vision functions)*	*during use case develop., testing and operation*
R45.7	*IKR*	*X*			*X*	*X*	*X*	*X*	*The selected platform*	*should*	*support application development in C++ and Python*	*during use case develop., testing and operation*
R45.8	*IKR*	*X*			*X*	*X*	*X*	*X*	*The selected platform*	*should*	*support video input (might be replaced by recorded videos)*	*during use case develop., testing and operation*
R45.9	*IKR*	*X*			*X*		*X*		*The selected platform*	*should*	*support Full Yolo V4 and YoloV7 that will be used as DL models for object detection. The goal is using YoloV7*	*during use case develop., testing and operation*
R45.10	*NAV*	*X*			*X*	*X*	*X*	*X*	*The selected platform*	*should*	*support faster I/O access via NVMe*	*during testing and operation*
R45.11	*AIKO*	*X*			*X*			*X*	*The selected platform*	*should*	*provide support to Floating Point execution*	*during testing and operation*
R45.12	*AIKO*	*X*			*X*			*X*	*The selected platform*	*should*	*provide DL Inference APIs for the available accelerators*	*during testing and operation*
R45.13	*BSC*	*X*	*X*		*X*	*X*	*X*	*X*	*The selected platform*	*should*	*provide adequate interference channels mitigation support (Partitioning, QoS, ...)*	*during testing and operation*
R5.1	BSC					X	X	X	Case studies	shall	provide a toy NN example excercising the main elements of the SAFEXPLAIN SW stack	during testing and operation
R5.2	IKR				X	X	X	X	The DL models deployment on the selected platform	shall	be imported into the target platform via PyTorch *.pt weights files or via ONNX.	during use case develop., testing and operation
R5.3	IKR						X		The DL models deployment on the selected platform	shall	achieve >75% accuracy on detection test set	during testing and operation
R5.4	IKR						X		The DL models deployment on the selected platform	shall	achieve >75% accuracy on distance estimation test set	during testing and operation
R5.5	IKR					X	X		The DL models deployment on the selected platform	shall	achieve 10 FPS frame rate for a 608x608 network size fort at least 1080p image resolution	during testing and operation
R5.6	NAV					X	X	X	Case studies	shall	Develop training dataset (possibily from simulation environment) to train the perception models	during use case development
R5.7	NAV					X	X	X	Case studies	shall	Create test set/scenarios (covering different challenging scenarios and edge cases) along with the ground truth for testing the developed models	during use case development
R5.8	NAV					X	X	X	Case studies	shall	Develop state-of-the art perception models which meets the performance and inference requirements of the use case	during testing and operation
R5.9	AIKO					X	X	X	Case studies	shall	be executable and testable on the platform	during use case development
R52.2	*AIKO*		*X*			*X*	*X*	*X*	*Case studies*	*shall*	*be evaluated by safety analysis tools*	*during testing and operation*
R52.1	*IKR*		*X*			*X*	*X*	*X*	*Case studies*	*shall*	*specify and implement safety functions based on DL software*	*during use case development*
R54.1	*BSC*	*X*			*X*	*X*	*X*	*X*	*Case studies*	*should*	*use the same or compatible software stack*	*during testing and operation*
R54.2	*BSC*	*X*			*X*	*X*	*X*	*X*	*Case studies*	*should*	*use the same HW platform*	*during testing and operation*

SC month 06				
m06				
R2.1, R2.2, R2.3	R2.1, R2.2, R2.3	Initial set of requirements, recommendations, ***architectural design patterns*** and procedures for (i) DL-software specification, design and implementation, and (ii) DL-software V&V	shall	be collected from selected standards
R2.10	R2.10	A common substrate of selected safety standards	shall	be identified
	R2.10	All partners	shall	select the safety standards that will be used as reference
R23.1	R23.1	The first draft of evaluation framework and metrics	shall	be prepared and agreed (WP2-3). Address the following dimensions: explainability, traceability, safety, robustness, confidence, failure rate, easy to implement safety patterns. Regarding explainability, allow determining whether explanations are sufficient for
R3.1	R3.1	AI library and the underneath framework on which it builds	shall	
R3.3	R3.3	The audience for human understandable representations (e.g., safety experts, end users, DL developers)	shall	be identified
R35.1	R35.1	All Use Cases (UCs)	shall	provide information about case studies architecture, components and dataflow
	R35.1	UC partners	shall	provide DL descriptions (layers, hyperparameters) that need explanation
R4.1	**R4.1**	The selected HW platform(s)	should	provide adequate observability channels (e.g Performance Monitoring Counters) on CPU, SoC and interconnect events
R4.2	**R4.2**	The selected HW platform(s) run-time	should	allow (not prevent) configuration of control registers for QoS and collection of hardware events
R4.3	**R4.3**	The selected HW platform(s)	should	provide debug support via IDE for Windows and Linux
R4.4	**R4.4**	The selected HW platform(s)	should	allow collecting end-to-end timing of selected libraries and applications
R45.1	R45.1	All partners	shall	have selected the target platform
	R45.1	BSC will lead and the rest of the partners	shall	contribute to the document describing the different boards considered
R45.2	R45.2	All partners	shall	have idenitfied the software support required on top of the target platform
R45.3	R45.3	All partners expecting to need a HW board	shall	have procured and setup the board according to WP1 specifications
R45.4	*R45.4*	The run-time of the selected HW platform	should	support PyTorch, TensorFlow and optionally Keras libraries
R45.5	*R45.5*	The run-time of the selected HW platform	should	support NVIDIA Jetpack, TensorRT, python Libraries
R45.6	*R45.6*	The run-time of the selected HW platform	should	support OpenCV libraries
R45.7	*R45.7*	The run-time of the selected HW platform	should	support C++ and Python applications
R45.8	*R45.8*	The run-time of the selected HW platform	should	support video inputs
R45.9	*R45.9*	The run-time of the selected HW platform	should	support the execution of Yolo (at least v4 but targeting v7)
R45.10	*R45.10*	The run-time of the selected HW platform	should	NVMe I/O access
R45.11	*R45.11*	The run-time of the selected HW platform	should	support FP operations
R45.12	*R45.12*	The run-time of the selected HW platform	should	provide DL inference API for the available accelerators
R45.13	*R45.13*	The run-time of the selected HW platform	should	provide mechanisms to contorl resource partitining, allocation, and QoS features
R5.1	R5.1	Use cases together with interested WP's	shall	define the requirements for the toy model to exercise safexplain sw stack
R52.1	R52.1	All UCs	shall	define safety goals and safety scenario together with wp2 for each use case
R54.1	R54.1	All partners	shall	have idenitfied the software support required on top of the target platform
	R54.1	All partners	should	have selected the same or compatible software stack on top of the target platform
R54.2	R54.1	All partners	shall	have selected the target platform
	R54.14	All partners	should	have selected the same target HW platform

SC month 18				
m18				
R2.1, R2.2, R2.3	R2.1, R2.2, R2.3	Initial ***safety assurance case templates*** for ***DL-usage level*** D & C	shall	be defined and provided together with a simple example
R2.4	R2.4	New methods for ***safety analysis*** that consider DL-software failure rate quantification	shall	be proposed
R2.5	R2.5	Initial safety ***architectural design patterns*** for safe DL deployment (HW and SW techniques) for DL-usage level D & C & A	shall	be defined and early evaluations done on the selected platform
R23.1	R23.1	WP2 and WP3	shall	consolidate the metrics of SCm06.3.1 to evaluate DL components
R3.3	R3.3	First set of XAI algorithms	shall	support different representations (text, tabular, graph, visual)
R3.4	R3.4	First set of XAI algorithms	shall	support both local and global explanations
R3.5	R3.5	DL components	shall	be specificable (data/model) in accordance with SCm18.2.1 and SCm18.2.2 specification for SP1
R3.6	R3.6	An online safety DNN architecture based on supervisors	shall	be designed to support online monitorability by allowing self-monitoring and self-diagnosis of DL in CAIS for SP1 (if needed), including outlier detection and types of input data missing in the training set
R3.9	R3.9	DL software implementations	shall	realize explainable AI architectures on the target platform for SP1
R3.10	R3.10	Explainable DL components	shall	be designed and prototyped for CNN and MLP types. Specifically for YOLO version that is selected by UCs
R3.11	R3.11, R3.12	A "XAI by design" architecture	shall	be designed to support explainability/traceability requirements resulted from WP2 for SP1 (Safety Pattern 1). The architecture should allow adoption by all UCs
R3.13	R3.13	Explainable DL components	shall	be measured and characterized by the evaluation metrics as defined in SCm18.2.1 and SCm18.2.2
R345.1	R345.1 R5.22	The use-case runnables	shall	be able to run on the testing platform
R35.3	R35.3	WP3	shall	provide requirements on case studies implementations, if any
R4.7	R4.7			
R4.8	R4.8	An initial identification of sources of timing interference in the platform	shall	be provided
	R4.8, R4.11	Platform-level support for controllabilty and predictability	shall	be identified
R4.9	R4.9	Evidence that HEMs can be read on the SAFEXPLAIN board	shall	be provided
	R4.8, R4.9, R4.11	An initial SW library for HEM configuration and control/mitigation of sources of timing interference	shall	be provided
R4.10	R4.10	At least one statistical timing analysis technique/method , compatible with SP1 requirements	shall	be developed
	R4.10	A baseline version of at least one statistical timing analysis tool, partially compatible with the adopted platform configuration	shall	be provided
R4.11	R4.11	The initial architectural safety patterns	shall	be assessed with respect their requirement on HW and SW platform l
R4.12	R4.12	Manual or partially automated support for collecting timing information from program execution	shall	be provided
R4.14	R4.14	A baseline version the platform monitor with standard monitoring capabilities (i.e., alive, deadline	shall	be provided
R4.15	R4.15	A baseline version of E2E message protection	shall	be provided
R4.16	R4.16	A baseline version of integrity protection for persistent data	shall	be provided
R4.17	R4.17	A baseline version of an extensible data visualization framework	shall	be provided
R4.18	R4.18	A proof-of-concept implementation of a the integration testing framework	shall	proposed
R4.19	R4.19	The platform	shall	be able to save and re-play use cases
R45.3	R45.3, R4.5	WP4	shall	define platform sw API, requirements for reproducibility and monitoring on case studies
R5.1	R5.1	WP5	shall	develop a toy NN example excercising the main elements of the SAFEXPLAIN SW stack and in line with SP1
R5.2	See 5.9			
R5.3	R5.3, R5.4, R5.5, R5.8	WP5 Use Cases (UCs)	shall	define performance goals/thresholds to be met for each case study
R5.4	see R5.3			
R5.5	see R5.3			
R5.6	R5.6, R5.7	WP5 Use Cases (UCs)	shall	collect training and testing datasets for each case study
R5.7	see R5.8			
R5.8	See R5.3			
R5.9	R5.9	WP5 Use Cases (UCs)	shall	stub each case study to work on the SAFEXPLAIN platform
	R5.9	WP5 Use Cases (UCs)	shall	complete 50% of the porting of the final case studies
R52.2	R52.2	WP2	shall	provide requirements on case studies implementations, if any

SC month 30				
m30				
R2.1, R2.2, R2.3, R2.4	R2.1, R2.2, R2.3, R2.4	Consolidated ***safety assurance case templates*** for DL-usage level D, C and A	shall	be defined and provided together with a simple example and agreed with WP3
R2.5	R2.5 R2.7	Consolidated safety ***architectural design patterns*** for safe DL deployment (HW and SW techniques) for DL-usage level D, C & A	shall	be tailored and evaluated on the selected platform
R2.6	R2.6	Consolidated safety ***architectural design patterns***	shall	consider mechanisms to reduce the impact of adversarial attacks
R2.7	see R2.5			
R2.8	R2.8	A ***safety concept*** for the railway case study	shall	be defined and assessed by external certification authority (e.g., TUV) and internal experts (EXI) (not necessarily positively)
R2.9	R2.9	Main project concepts and outcomes applied to the case studies	shall	be ready for review by domain experts (e.g., EXI)
R23.1	R23.1, R23.2	Common set of definitions and compliance means	shall	be finalized and agreed with WP2 to achieve safety goals
R3.3	**R3.3, R3.4, R3.9**	DL software implementations	shall	realize explainable AI architectures in SCm30.3.6 on the target platform used by the case studies
R3.4	**see R3.3**			
R3.5	**R3.5**	Explainable AI architectures	shall	be realized in accordance with SCm30.3.1, SCm30.3.2, SCm30.3.3 and SCm30.3.4 specification and design principles for all safety patterns
	R3.5	DL component specifications	shall	be finalized and agreed with WP2 to achieve safety goals
R3.6	**R3.6**	An online safety supervision architecture based	shall	be prototyped to support online monitorability by allowing self-monitoring and self-diagnosis of DL in CAIS for all safety patterns, including outlier detection and types of input data missing in the training set
R3.7	**R3.7**	Datasets used in UCs	shall	be evaluated by the defined data explanation metrics
R3.8	**R3.8**	Approach to improve datasets	shall	be proposed (example: GAN) aiming for improving the data evaluation results
R3.9	**see R3.3**			
R3.10	**R3.10**	Explainable DL components	shall	be designed and prototyped for other DL types (RNN/LSTM, transformers)
R3.11	R3.11, R3.12	"XAI by design" architecture(s)	shall	support explainability/traceability requirements resulted from WP2 for all safety patterns
R3.12	see R3.11			
R3.14	R3.14	A proof-of-concept for how to specify DNNs as a collection of explainable modules together with data/model specification	shall	be developed
R345.1	R345.1 R5.22	The platform	shall	provide containerized support to deploy the use-case applications
	R345.1 R5.11 R45.3, R4.5, R4.6, R4.18	The use-cases	shall	be fully ported and integrated on testing platform and meet platform sw API, requirements for reproducibility and monitoring
R35.2	R35.2	**The evaluation metrics**	**shall**	**be measured for all UCs**
R4.5	**R345.1**	The use-cases	shall	be fully ported and integrated on testing platform and meet platform sw API, requirements for reproducibility and monitoring
R4.6	**R345.1**	The use-cases	shall	be fully ported and integrated on testing platform and meet platform sw API, requirements for reproducibility and monitoring
R4.7	R4.7	A initial version of the platform monitor supporting the collciton of SAFEXPLAIN custom metrics	shall	be provided
R4.8	R4.8	A consolidated analysis of sources of timing interference in the adopted platform configuration	shall	be provided
	R4.8, R4.11	Platform-level configuration to support Safety Pattern requirements on controllabilty, predictability, and performance	shall	be provided
	R4.8, R4.9, R4.11	A consolidated SW library for HEM configuration and control/mitigation of sources of timing interference in the platform	shall	be provided
R4.9	see R4.8			
	see R4.8			
R4.10	R4.10	A consolidated version of at least one statistical timing analysis technique/method , compatible with Safety Patterns requirements	shall	be provided
	R4.10	At least one statistical timing analysis tool, compatible with the adopted platform configuration	shall	be consolidated
R4.11	R4.11	The proposed DL safety patterns in	shall	be concretely mapped to HW ans SW platform level solutions
R4.12	R4.12	Support for partially automated collection of timing information from program execution	shall	be provided
R4.14	R4.14	A monitoring concept	shall	be proposed
	R4.14	A testing concept	shall	be proposed
R4.15	R4.15	E2E message protection	shall	be finalized
R4.16	R4.16	Integrity protection for persistent data	shall	be finalized
R4.17	R4.17	The data visualization framework	shall	be consolidated based on use case needs (if applicable)
R4.18	R4.18	The platform monitor	shall	be extended with metrics and techniques, identified by WP2 and WP3
	R4.18	A baseline version of utilities for testing and test-automation	shall	provided, based on the outcomes of WP2 and WP3
R5.1	R5.1 to R5.5 + 5.9	The use-cases	shall	be fully ported and integrated on testing platform and meet platform sw requirements for reproducibility and monitoring
R5.2	see R5.1			
R5.3	see R5.1			
R5.4	see R5.1			
R5.5	see R5.1			
R5.9	see R5.1	The use-cases	shall	be fully ported and integrated on testing platform and meet platform sw requirements for reproducibility and monitoring
R52.2	R52.2, 235.3	The use-cases	shall	be analysed by safexplain sw stack on the aspects of functional safety and explainability

SC month 36					
m36					
R2.1, R2.2, R2.3, R2.4, R2.5, R2.6, R2.7	R2.1, R2.2, R2.3, R2.4, R2.5, R2.6, R2.7		Safety guidelines and arguments for DL software adoption in the critical domain	shall	be described
R2.8	R2.8		The safety concept in SCm30.2.2	shall	be positively assessed by certification experts after taking into consideration their recommendations
R2.9	R2.9		Recommendations for future standard improvement and extension to admit guidelines and arguments in SCm36.2.2	shall	be consolidated
R3.2	R3.2		FUSA-aware DL library implementations	shall	reach 75% performace of the non-FUSA aware libraries performance (e.g. frames per second for camera-based object detection) for the automotive case study
	R3.2		FUSA-aware DL library implementations	shall	realize explainable AI architectures in SCm30.3.6 on the target platform used by the case studies, and be integrated in the case studies
R4.7	R4.7		The platform monitor	shall	support a relevant set of SAFEXPLAIN custom metrics, with special focus on those related to understandability and explainability.
R4.8	R4.8, R4.11		Platform-level configuration to support Safety Pattern requirements on controllabilty, predictability, and performance	shall	be updated to accommodate emerging requirements and refinements, if any.
	R4.8, R4.9, R4.11		SW library for HEM configuration and control/mitigation of sources of timing interference in the platform	shall	be updated to accommodate emerging requirements and refinements, if any.
R4.10	R4.10		Statistical timing analysis methods and tools, compatible with the adopted platform configuration,	shall	be updated to accommodate emerging requirements and refinements, if any.
R4.11	R4.11		The mapping of safety patterns to HW ans SW platform level solutions	shall	be updated to accommodate emerging requirements and refinements, if any
R4.12	R4.12		Support for partially automated collection of timing information from	shall	be updated to accommodate emerging requirements and refinements, if any.
R4.13	R4.13		A final set of guidelines and practices for statistical predictability	shall	be provided
R4.14	R4.14		The monitoring concept	shall	be finalized
	R4.14		The platform monitor	shall	implement the monitoring concept
R4.18	R4.18		The test-automation framework	shall	be finalized
	R4.18		The testing concept	shall	implemented and integrated in the test-automation framework
	R4.18		The testing concept	shall	be finalized
R5.3	R5.3		FUSA-aware DL library implementations	shall	reach 75% performace of the non-FUSA aware libraries performance (e.g. frames per second for camera-based object detection) for the railway case study
R52.2	and R2.9 R52.2.				