D2.3 Safety Concept

Version 1.0

Documentation Information

Contract Number	101069595
Project Website	www.safexplain.eu
Contratual Deadline	31.03.2025
Dissemination Level	PU
Nature	R
Authors	Javier Fernández (IKR), Giuseppe Nicosia (EXI), Francesca Guerrini (EXI), Stefano Lodico (EXI)
Contributors	Irune Yarza (IKR), Irune Agirre (IKR), Carlo Donzella (EXI)
Reviewed by	Enrico Mezzetti (BSC)
Keywords	AI, Functional Safety, V&V, Catalog Scenarios, Safety Concept

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	Second draft after addressing internal reviewer's comments
V1.0	First consoldiated version

Table of Contents

Executive Summary

This document includes an update of D2.1 and D2.2, incorporating enhancements and additions to the previously established Deep Learning (DL) software Verification and Validation (V&V) strategy (deliverable D2.1) and the development of a safety concept for a railway case study.
This safety concept aims to assess the previously defined strategies and specific solutions for implementing safety patterns in safety-critical software systems that incorporate Artificial Intelligence (AI) components (previously included in deliverable D2.2). The architectural solutions proposed in the railway case study seek to identify potential inconsistencies in the approach, address any contradictions with the requirements of the Bases of Assessment, and evaluate the feasibility of applying the approach within the given context.

1. Introduction

This document compiles the results from all tasks developed in Work Package (WP) 2: T2.1, T2.2, T2.3, T2.4, and T2.5. The objective of these tasks is to define and evaluate functionalities based on DL components that are safety-related or whose use may have implications for safety functions.

Specifically, this deliverable establishes a safety concept for the railway use case, following the lifecycle developed in the SAFEXPLAIN project (T2.1, deliverable D2.1) and including relevant safety architectural design patterns for different DL usage levels (T2.3, deliverable D2.2). This safety concept serves as a foundation for validation by a certification authority, supporting future industrialization. Additionally, this deliverable expands the previously defined V&V strategy (task T2.2, deliverable D2.1). The results will be used to validate the railway case study approach, which is a key task within WP2. Instead of partially sharing the content of T2.4, we reserve part of it for deliverable D2.5, where the final progress on this task will be presented.

Since this deliverable updates and completes the content previously introduced in deliverable D2.1 and deliverable D2.2, we aimed to minimize redundancy across deliverables by avoiding repeating information whenever possible. Therefore, we refer the reader to those deliverables for background concepts and foundational aspects in case of any doubts.

This document is organized as follows:

- Section 0 introduces the railway case study and outlines the activities undertaken to analyze the feasibility of applying the proposed strategies and specific solutions for implementing safety patterns tailored to the railway domain. These efforts aim to assess their alignment with safety certification requirements through the development of a safety concept. Finally, this section presents the methodology followed for the safety assessment and the results obtained.
- Section 3 extend the V&V strategy previously introduced in deliverable D2.1 at vehicle and element level. To that end, at vehicle level this section defines a set of example scenarios related to the three use cases employed in SAFEXPLAIN project: railway (led by IKERLAN), automotive (led by EXIDA) and aerospace (Led by AIKO). Additionally, at element level this section analyses the triggering conditions derived through System-Theoretic Process Analysis (STPA), still for the three use cases. Finally, it includes examples of V&V strategy application in the three use cases defined in SAFEXPLAIN project.

2. Safety Concept

The main result of this document is the definition of the Safety Concept for railway case-study based on the SAFEXPLAIN contribution, and the documentation of the feedback provided by the certification authority.

The definition of the Safety Concept will serve to illustrate a practical application of the lifecycle procedure developed around the use of AI in safety domains. As outputs of the work regarding SAFEXPLAIN:

- The Railway Safety Concept is included in Annex A and is structured into three parts. The first part provides an informative introduction outlining the SAFEXPLAIN project and its main topics. The second part describes the reference safety architecture and safety patterns submitted for review by TÜV Rheinland, adapted and extended from deliverable D2.2. Finally, the third part defines the system concept specification of a railway domain object detector controller. This section follows an incremental strategy, progressing from the lowest DL usage level (Usage Level D / EASA Level 1) to the highest one (DL Usage Level A1 / EASA Level 3). It presents a set of safety patterns, proposing specific system architectures, safety techniques, and their application on the project's target platform (NVIDIA Orin).
- The presentation and the main open issues and comments discussed during the review meeting with TÜV Rheinland certification authority is provided in the Annex B. Additionally, the methodology followed in the assessment, as well as a summary of the main points discussed during the meeting, are reported.

2.1. Safe technical assessment and expert certification review

This subsection is related to Task T2.5, which is scheduled to run from month 13 to month 36. This task consists of two key activities: one focused on the AI-FSM and the other on the railway safety concept definition. The former was covered in deliverable 2.1, while the latter is addressed in this subsection. We have maintained the same methodology as explained in the deliverable D2.1, which is summarized in the Figure 1.

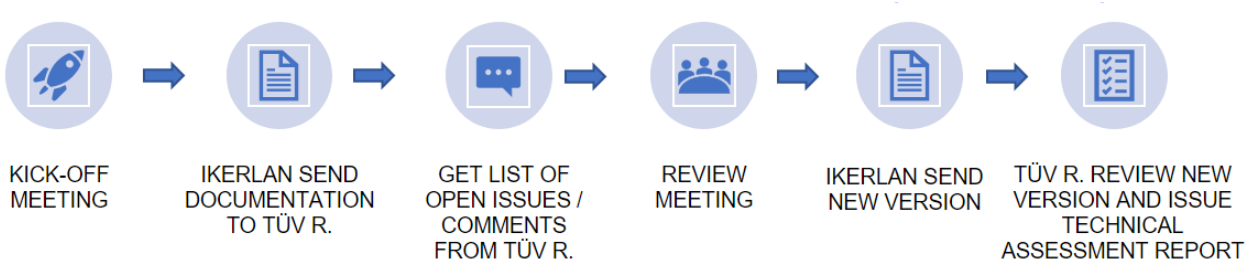


Figure 1. Safety concept review steps

We have included in the “Annex A: Railway Safety Concept” the same documentation sent to TÜV Rheinland and in the “Annex B: Review meeting presentation” we have included the review meeting presentation in which we discussed the main open issues and comments received from the first TÜV Rheinland assessment. The review meeting focused on information exchange and experience sharing related to these topics. At the time of writing of this deliverable, the TÜV Rheinland assessment have not been received yet. However, we can share the general impression, and the main points discussed during the review meeting and, once the final assessment is received, we will include its conclusions in deliverable D2.5.

One of the points discussed was related to AI risk factors and the complexity of identifying certain systematic faults during the design and V&V processes. In fact, we agreed that, due to the extremely high number of parameter variations in an AI model, these faults may not be easily detectable. However, they should not be considered random, but rather as systematic errors arising from those pseudorandom distributions that are difficult to detect.

We discussed the importance of the decision function, which brings together the redundant channels and therefore it shall have more stringent requirement than those posed over the AI based SW components it gets the input from. They requested to be more specific about the explainability techniques that can be implemented in the supervision vision.

We discussed how the properties we identified for AI technology align with the clauses of ISO/IEC TR 5469. In this context, they proposed including clause 8.4.3, "Issues Related to Learning from the Environment," and explicitly mentioning the other relevant clauses.

From the system description, TÜV Rheinland expressed concerns about relying solely on cameras instead of incorporating diverse sensor technologies such as LIDAR or infrared cameras. We explained that this railway safety concept initially focuses on camera-based AI; however, our approach can be extended to include additional input technologies in future development phases.

Finally, as general impression, they conclude that we have provided a reasonable approach for an architectural solution tailored to a railway case study. In the same way, they conclude that our work outlines key strategies and tailored solutions for implementing safety patterns in safety-critical software systems that incorporate AI components.

Additionally, since from some AI-based parts the maximum Safety Integrity Level (SIL) is 3 or 4, they remarked that all hardware and software for this AI components shall fulfil the requirements imposed by traditional functional safety standards for the corresponding SIL. With a small joke, they were confident that IKERLAN would take it into account, given that we had considered the EASA documentation, where a system consists of a traditional system based on functional safety and another containing the AI component.

Additionally, we presented *safeYOLO*, a statically allocated memory adaptation of YOLOv4[1]. This implementation focuses on adapting the inference phase of YOLOv4 to comply with MISRA C, a widely used coding guideline. We discussed that this serves as a strong starting point for aligning the software implementation of AI-based systems with traditional functional safety standards. However, TÜV emphasized the need to also address insufficiencies related to the model itself, particularly those stemming from the training phase.

[1] https://github.com/AlexeyAB/darknet

3. DL Safety Lifecycle for DL-software V&V

Figure 2, depicts the V&V strategy followed for each of the three use cases in the automotive, railway and aerospace domains deployed within the SAFEXPLAIN project.

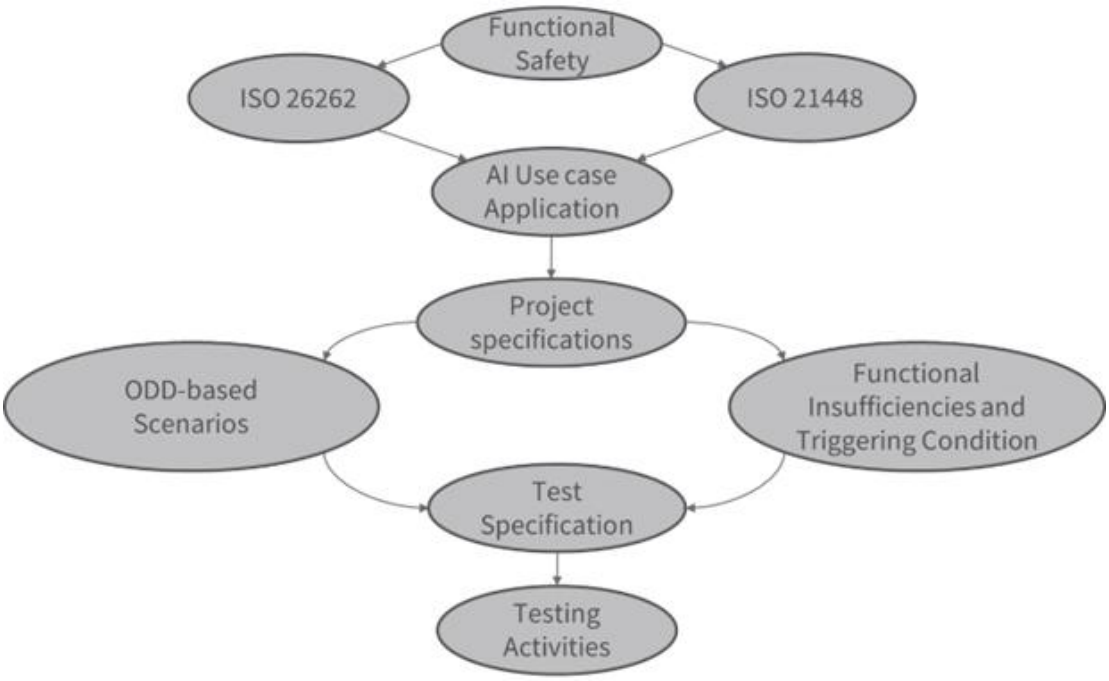


Figure 2. V&V Strategy.

The adopted V&V strategy and the associated methods for the ML/DL-V&V extends the traditional FuSa approach by addressing also "*hazards resulting from functional insufficiencies*" as it was explained in the deliverable D2.1. Our V&V strategy incorporates testing techniques based on those provided in ISO 26262 [1] standard and ISO 21448:2022[2], aiming to stablish a valid method for verifying the intended functionalities at vehicle level (scenario-based) and at component level (triggering condition-based).

Since this deliverable is an extension of deliverable D2.1, and to facilitate the reader's understanding, we have included relevant content from that document regarding the test methods used in the V&V strategy related to both ISO 26262 and ISO 21448:

- ISO 21448 (testing activities are focused on the scenarios):
  - Analysis of environmental conditions and operational use cases (ISO 26262-4 Table 3 method 1h; ISO 21448 Table 6 Method H)
  - Analysis of triggering conditions (Method N, Table 6)
- ISO 26262 (testing activities are focused on proving the safety requirements implementation and performance of safety mechanism):
  - Requirements-based test (Method 1a - ISO 26262-4 table 13) Fault injection test (Method 1b - ISO 26262-4 table 13; Method 1d - ISO 26262-4 table 14)
  - Fault injection test for triggering condition-based testing (ISO 26262-4 Table 4 method 1b; ISO 21448 Table 6 Method N)
  - Long-term test (Method 1c - ISO 26262-4 table 13; Method 1b - ISO 26262-4 table 14; Method 1d - ISO 26262-4 table 16)
  - Performance test (Method 1a - ISO 26262-4 table 14)

The main goal of the Verification & Validation (V&V) strategy is to:

- Find the hazardous scenarios related to the design
- Provide the evidence (e.g., test reports, …) to prove the following:
  - the Sense-Plan-Act elements (sensors and processing/decision algorithm) provide their proper functionality correctly
  - the robustness of the system or functionality against the triggering condition
  - the absence of unreasonable risk due to hazardous behaviour of the intended functionality
  - the achievement of an acceptable risk level.

V&V strategy is applied both at vehicle level and component level:

- At vehicle level, a list of scenarios (Scenario Catalogue) has been defined which describes the intended functionality in real-life situations for each use case domain. From the catalogue of the scenarios the related test cases were derived to demonstrate, by following the steps defined in the test cases, that the intended functionality is behaving as specified and the residual risk meets the acceptance criteria (criteria which represent the absence of an unreasonable level of risk).
- A test matrix is derived from test specifications which will be executed, guided by the test matrix, to validated and ensures that the functionality works as expected. This test matrix will delineate the constraints and every value that should be set for each test step (e.g. ego vehicle, target vehicle, environment...). The test matrix also facilitates the documentation of the outcomes of each test conducted, to generate a descriptive report at the conclusion of the test activity.
- At element level, we have analysed the design through System-Theoretic Process Analysis (STPA) to:
  - Identify the most critical triggering condition related to the elements design.
  - Identify the missing safety mechanism (if any) to cover the triggering condition and bring the harm to an acceptable level implementing the needed functional modifications, or
  - Define the test cases to test the safety mechanisms already implemented.

By following the steps defined in the test cases, the effectiveness of the implemented safety mechanisms is proved in identifying the system's weaknesses. Additionally, it is demonstrated that the intended functionality behaves as specified.

3.1. Scenario catalogue and related test matrix

The purpose of the scenario catalogue is to define the set of scenarios to describe how the intended functionality works when a specific relevant situation occurs. The scenario catalogue has been defined for each of the SAFEXPLAIN project use case domains: Automotive, Railway and Aerospace.

In every scenario, to enable the simulation for the testing activities, the following relevant conditions and constraints are specified:

- Conditions and/or constraints related to the ego vehicle (e.g., ego vehicle speed, ego vehicle acceleration, offset w.r.t. the target, initial distance from the target, ego vehicle status…).
- Conditions and/or constraints related to the vehicle identified as target (e.g., target vehicle speed, target vehicle acceleration, vehicle status…).
- Conditions/Constrains related to environment (e.g., luminosity, temperature…)

- Vehicles and Scenarios Pre-conditions (e.g., ego vehicles speed, ego vehicles path, override conditions, ….).

Depending on the use case different conditions and/or constrains have been reported, for example:

- For automotive:
  - Steering, as well as yaw rate, could have a critical impact if not regulated, given their relevant exposure; therefore, thresholds were specified. Since steering and yaw are not applicable in railway and aerospace domains, thresholds for these domains were not specified.
  - Road surface condition could have critical impact (unlike railway/aerospace), therefore the road surface condition is needed to be defined as the most common street in a good condition (asphalt and concrete).
- For Railway:
  - Speed limit, when the train goes in/out the train station and when goes outside the train stations.
  - Environmental condition:
    - Unlike the automotive use case, this domain considers not only sunny days but also cloudy and rainy days, in accordance with the safety concept.
    - In the automotive, scenarios in the night were also considered, but here the scenarios created are just in the daytime, since the night was excluded in the safety concept of the use case.
- For Aerospace:
  - Since in the space no object can be stationary, we consider the target as reference frame, and we defined just the relative speed of the ego vehicle w.r.t. the target vehicle
  - In the environmental conditions, since we are not in the earth:
    - There is a different gravity acceleration which was assumed.
    - The orbits where the ego vehicle flies are also specified (LEO/GEO).
    - Since concepts of “day and night” is not applicable in the space, luminosity is defined not as a range of “intensity of pixels in the image”.
    - Since we are in space, weather conditions are not applicable, so excluded.

Each scenario created is based on our everyday life experiences (if applicable) and the Operational Design Domain (ODD) where the constrains/conditions (by design) related to the functionality are defined.

The criteria to choose a scenario, is based on what the related test and analyses could reveal or confirm, as well the frequency with which the specific scenario could occur. A scenario that is rare is not worth analysing.

Indeed, only for the automotive domain, to quantify the probability of occurrence of a scenario (e.g., every 800 h…), we based on the VDA-702[1, 3], published by the German Association of the Automotive Industry, which describes every specific relevant situation and their exposure parameters (based on frequency and duration). Basically, it gives some “marks” to each situation and basing on these marks and the constrains derived from it (defined in the document itself) we defined the probability of occurrence.

In order to verify whether the intended functionality works as expected, every defined scenario must be validated through related testing activities which is regulated by related test case

specification which describes the test activity to be performed step by step, with constrains/conditions related to the specific step described.

These tests are necessary to analyse the functionality behaviour in that specific scenario and in those specific conditions. At the end of each testing activity, the outputs shall be evaluated to determine whether the acceptance criteria are met.

In order to manage and facilitate every testing activity, a test matrix derived from the test cases specification has been defined.

As it was previously outlined, the test matrix is a useful tool that maps every test case and let the tester to be guided in the testing activities. Its purpose focuses on:

- Indicate visually and in an easily comprehensible manner the constraints and conditions of the ego vehicle, as well as those of the environment, target vehicle and every value to be set in every test case.
- Describe how many repetitions of the tests shall be performed.
- Collect and assess the test result ensure a comprehensive report of the testing activity.

3.2. Focus STPA (triggering condition) and related test matrix

In order to identify potential specification insufficiencies, potential performance insufficiencies and potential triggering conditions leading to a hazardous behaviour a SOTIF analysis has been conducted by using STPA method.

STPA is an analysis of the hazards that could occur in a system, starting from the assumption that hazardous behaviour can also be caused even when no component may have failed. STPA is very useful to optimize the hazard/risk analysis techniques to guarantee that the intended functionality, also if the system is very complex, works correctly.

STPA analysis is developed in 4 steps:

1. Defining the purpose and scope of the analysis: Here is necessary to identify the losses to be prevented by analysing the design, which are vehicle-level states or conditions, with particular other conditions, will lead to a loss (harm). From the design safety critical hazard must be also derived and condition/constraints as well.
2. Modelling of the control structure: Here all the specifications are analysed to identify a list of control actions (CAs) of the system, which are how the intended functionality is expected to control that hazardous event.
3. Identification of unsafe control actions: Here we identify the Unsafe Control Actions (UCAs), which are actions (associated to the Hazard Analysis and Risk Assessment (HARA)) that, in a particular context and worst-case environment, will lead to a vehicle-level hazard. To ensure the UCAs are prevented, controller safety constraints (SCs) can be defined, specifying assertions on the controller behaviours that need to be satisfied to prevent UCAs from occurring.
4. Identification of causal scenarios: the final core step of STPA identifies the functional insufficiencies and triggering condition. The former refers to the limitation of the functionality caused by the triggering condition, while triggering conditions are the factors that initiate a subsequent system reaction leading to hazards. A triggering condition can be anything that causes a limitation of the functionality, such as camera of the sensor blinded by newspaper or signal corrupted by electromagnetic interference.

The purpose of this analysis is to identify the improvements that need to be implemented to fulfil the safety constraints identified in the STPA. The triggering condition identified via STPA analysis shall be validated through dedicated test cases, which shall:

- Include an ID
- Include a brief description to describe the aim of the test itself
- Report step by step all the pertinent conditions and constraints to provide a comprehensive guide for the tester.

The validation of the triggering condition and the STPA analysis is contingent upon the successful outcome of each test case, with a test case being deemed successful if every step in it is successfully executed.

3.3. Example of V&V strategy application

This subsection presents the implementation of our proposed V&V strategy for the three use cases addressed in the SAFEXPLAIN project: automotive (led by NAVINFO), railway (IKERLAN), and aerospace (AIKO). To do so, this subsection follows the structure below for each of the previously mentioned use cases:

1. A description of scenarios extracted from the scenario catalogue (adapted to the three use cases), including their conditions/constraints.
2. A description of test cases at vehicle level and the expected behaviour at vehicle, sense, plan and actuator levels.
3. An example of triggering conditions derived from the previously identified hazards at vehicle level, including the associated control action, functional insufficiency and control action failure mode[2].
4. An example of the related test case will be provided to validate the corresponding triggering condition (each triggering condition shall be validated by a related test case).

[2] This document presents a reduced version of the complete set of triggering conditions and related test cases development by EXIDA. The whole information has not been fully shared because of EXIDA's intellectually property restriction.

3.3.1. Automotive Use Case

This subsection focuses on the automotive use case developed by NAVINFO. It shall be noted that the content related to the scenario catalogue and the vehicle level test case in the automotive use case was previously reported in the D2.1 deliverable; however, it has been included again for completeness, ensuring that examples for all SAFEXPLAIN applicable use cases are collected in one place.

3.3.1.1. Example of Scenario from the scenario catalogue

The scenario provided in this deliverable represents a vehicle driving following a target vehicle on highway, as depicted in Figure 3 (previously included in deliverable D5.1) . When the distance with the target vehicle decreases so that the driver is in dangerous zone (possible collision) the intended functionality shall warn the driver and, if no driver reaction occurs and the collision is imminent, shall decelerate the vehicle.

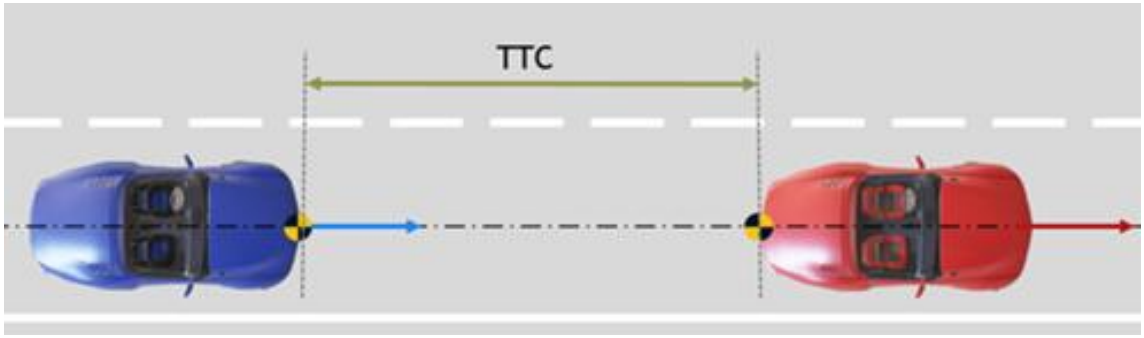


Figure 3. Visual representation of the scenario example

The scenario conditions/constraints are the followings:

- The Ego vehicle (depicted in red in the Figure 3) drives with a longitudinal acceleration lower than 2 m/s^2 towards a moving target vehicle (depicted in blue in the Figure 3) and is at a distance corresponding to a Time To Collision (TTC) of at least 4 s.
- The Ego vehicle speed range is [50 km/h, 130 km/h]
- The target vehicle drives at 80 km/h
- The following environmental conditions shall be present:
  - Dry and daylight with minimum 1000 lux and Sun angle >15° to horizon
  - Dry and night with maximum 10 lux
- Road surface is asphalt or concrete.
- The following pre-conditions shall be respected:
  - Both vehicles shall keep steady speed and path.
  - Steering angle shall be lower than the override threshold.
  - Yaw rate shall be lower than the override threshold.
- The probability of exposure (duration) of these scenario conditions is E2, considering the following combinations:
  - Driving behind other vehicle with normal distance – E4 (>10 % of average operating time): E.g., 10% of 8000h = 800 h
  - Driving with normal longitudinal acceleration (<2m/s2) – E4 (>10 % of average operating time): E.g., 10% of 8000h = 800 h
  - Driving in Highway– E4 (>10 % of average operating time): E.g., 10% of 8000h = 800 h

3.3.1.2. Example of Vehicle level test case

The following intended functionality capabilities shall be demonstrated:

3.3.1.2.1. *Step 1.* Track the red target vehicle and evaluate it as no-collision relevant.

Figure 4. Vehicle level test case Step 1

Pass/Fail criteria:

- **Vehicle level:**
  - Warning = It is not expected the provision of any warning to the driver.
  - Braking = It is not expected the provision of braking intervention.
- **Sense level:**
  - It is expected that the object is being detected and classified as a Car.
- **Logic level:**
  - It is expected that the Object, considering the safety distance between the ego-vehicle and the target vehicle, is being evaluated as “no-collision” relevant.
- **Actuator level:**
  - Warning = It is not expected the provision of any warning to the driver.
  - Braking = It is not expected the provision of braking intervention.

3.3.1.2.2. ***Step 2***. *When the distance, between* **the ego vehicle and the red target vehicle, is equal to the Time To Warning (TTW), the intended functionality shall evaluate the red target vehicle as collision relevant and provide at least 0,8 s before the start of the emergency braking the visual and audible warning to the driver (UN Regulation N° 152 clause 5.2.1.1, 5.5.1).**

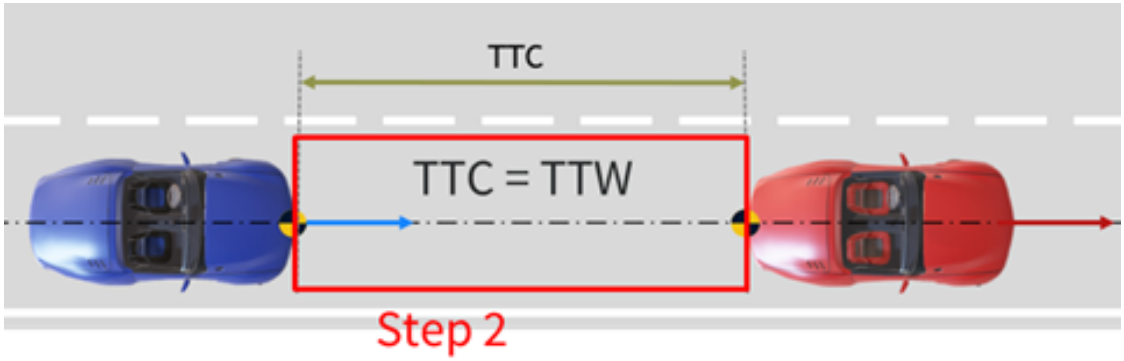


Figure 5. Vehicle level test case Step 2

Pass/Fail criteria:

- Vehicle level:
  - Warning = It is expected the provision, at least 0,8 s before the start of the emergency braking according to UN Regulation N° 152 [13], of audible and visual warning to the driver.
  - Braking = It is not expected the provision of braking intervention.
- Sense level:
  - It is expected that the object is being detected and classified as a Car.
- Logic level:
  - It is expected that the Object, considering that the safety distance between the ego-vehicle and the target vehicle is equal to TTW, is being evaluated as “collision” relevant.
- Actuator level:
  - Warning = It is expected the provision, at least 0,8 s before the start of the emergency braking according to UN Regulation N° 152 [13], of audible and visual warning to the driver.
  - Braking = For this step it is not expected the provision of braking intervention.

  Note: UN Regulation N° 152 [13] is the Regulation applicable for the approval of vehicles of Category M1 and N1 with regard to an on-board system to:

    - Avoid or mitigate the severity of a rear-end in lane collision with a passenger car,
    - Avoid or mitigate the severity of an impact with a pedestrian.

3.3.1.2.3. ***Step 3*. When the distance, between the ego vehicle and the red target vehicle, is equal to the Time To Collision AEB (TTC AEB), the intended functionality shall, if no driver reaction occurs, shall decelerate the vehicle providing at least 5.0 m/s2 (UN Regulation N° 152 clause 5.2.1.2).**

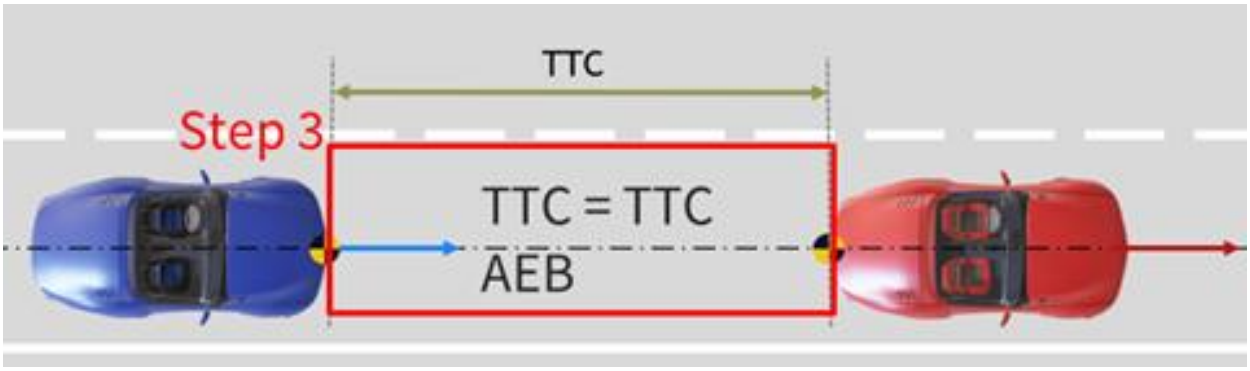


Figure 6. Vehicle level test case Step 3

Pass/Fail criteria:

- Vehicle level:
  - Warning = It is expected the provision, at least 0,8 s before the start of the emergency braking according to UN Regulation N° 152 [13], of audible and visual warning to the driver.
  - Braking = It is expected a deceleration of at least 5 m/s^2, according to UN Regulation N° 152 [13].
- Sense level:
  - It is expected that the object is being detected and classified as a Car.
- Logic level:
  - It is expected that the Object, considering that the safety distance between the ego-vehicle and the target vehicle is equal to TTC AEB, is being evaluated as “collision” relevant.
- Actuator level:
  - It is expected the provision, at least 0,8 s before the start of the emergency braking according to UN Regulation N° 152 [13], of audible and visual warning to the driver.
  - It is expected a deceleration of at least 5 m/s^2, according to UN Regulation N° 152 [13].

3.3.1.3. *From Hazard Vehicle Level to Triggering condition*

- Hazard Vehicle Level:
  - AEB provides emergency braking when not needed
- Control Action:
  - Perform emergency braking
- Control Action Failure Mode:
  - Emergency braking is performed when the collision is not imminent
- Functional Insufficiency:
  - The AEB reject an abortion request
- Triggering Condition:
  - Abortion rejected by inaccurate throttle signal due to low communication buses performances (e.g. busy buses, inadequate message priority or arbitration or EMI)
  - Abortion rejected by inaccurate steering signal due to low communication buses performances (e.g. busy buses, inadequate message priority or arbitration or EMI)
  - Abortion rejected by wrong algorithm elaboration (wrongly evaluates the brake pedal status, the speed reduction reached by the vehicle, lateral acceleration value, yaw rate).

3.3.1.4. *Example test case related to the triggering condition*

- Test ID: TCTC-01
- Tested Triggering Condition ID: TC2-0016
- Test Description: The test aims to verify whether the AEB is deactivated against a corrupted throttle signal due to low communication buses performance event.
- Test Steps:
  - ***Step 1*:**
    - Initial state:
      - Kl.15 = on
      - No warning message available
      - Intended functionality state: active
    - Operating Element:
      - Throttle Sensor System
      - CAN Bus Simulator
      - AEB ECU
    - Operating:
      - Sent valid throttle signal in time
    - Expected Result:
      - No warning message provided
      - Intended functionality state: active
  - ***Step 2:***
    - Initial state:
      - Kl.15 = on
      - No warning message available
      - Intended functionality state: active
    - Operating Element:
      - Throttle Sensor System
      - CAN Bus Simulator

        - AEB ECU
    - Operating:
        - Sent valid throttle signal in time
        - Injected Emergency Braking Request
    - Expected Result:
        - Emergency Braking warning provided in time according to intended functionality requirement
        - Emergency Braking Request provided in time according to intended functionality requirement
        - Intended functionality state: active intervening
- ***Step 3:***
    - Initial state:
        - Kl.15 = on
        - Emergency Braking warning provided
        - Emergency Braking Request provided
        - Intended functionality state: active intervening
    - Operating Element:
        - Throttle Sensor System
        - CAN Bus Simulator
        - AEB ECU
    - Operating:
        - After x ms it shall be sent a delayed throttle signal
    - Expected Result:
        - Intervention of dedicated CAN communication safety mechanism detecting the delay of throttle signal causing the deactivation of the functionality within x ms.
        - Intended functionality state: deactivated.

3.3.2. Railway Use Case

This subsection implements the V&V strategy on an use case in the railway domain.

3.3.2.1. Example of Scenario from the scenario catalogue

In this subsection is reported an example of one of the scenarios included in the scenario catalogue adapted to the Railway use case developed by IKERLAN. It represents a train that travels towards a static pedestrian on the tracks, as depicted in Figure 7.

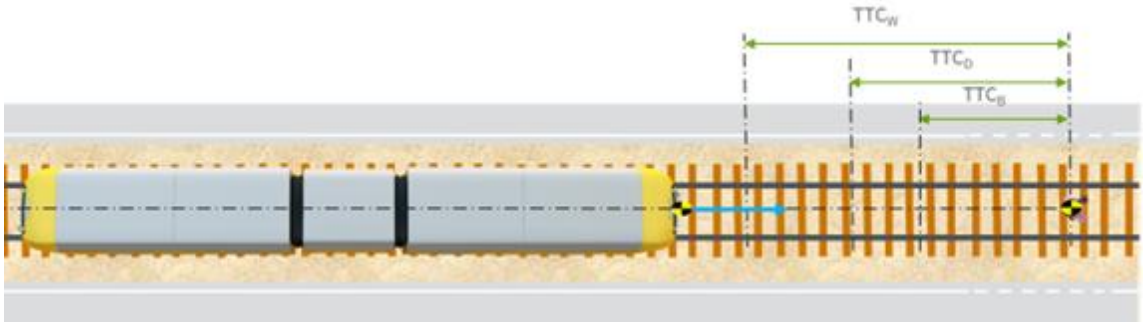


Figure 7. Visual representation of the scenario example

When the distance with the pedestrian decreases so that the collision is possible, the intended functionality shall react as follow:

- If the distance with the pedestrian is equal to Time To Warning (TTC_W) [1001 m, 1500 m], the intended functionality shall warn the driver about the upcoming collision.
- If the distance with the pedestrian is equal to Time To Deceleration (TTC_D) [701 m, 1000 m], the intended functionality shall reduce the train speed through service brake intervention.
- If the distance with the pedestrian is lower to Time To Brake (TTC_B) [700 m], the driver shall stop the train through emergency brake intervention.

The scenario conditions/constraints are the following:

- The train drives with a longitudinal acceleration lower than 1m/s^2 towards a pedestrian in the track and is at a distance of [0 m, 1500 m]
- The Train speed range is [0 km/h, 90 km/h]
- The speed limit in case the train goes in the station or out from the station shall be 30 km/h
- The speed limit in case the train is travelling outside the station shall be 90 km/h
- The pedestrian is stationary on the track
- The initial longitudinal offset is: 2000 m
- The maximum deceleration is: 0,8 m/s^2
- The following environmental conditions shall be present:
  - Dry and daylight with 1500 lx as minimum luminosity range and Sun angle >15° to horizon
  - Rainy day with [0,1 mm/h, 7,6 mm/h] as rain intensity threshold and with [1000 lx, 1499 lx] as luminosity range
  - Cloudy day with [1000 lx, 1499 lx] as luminosity range
- The following Pre-conditions shall be respected:
  - Train shall keep steady speed
  - no internal failure shall be present

3.3.2.2. *Example of Vehicle level test case*

The following intended functionality capabilities shall be demonstrated:

3.3.2.2.1. ***Step 1*. The train runs at constant speed on the track in the direction of the pedestrian.**

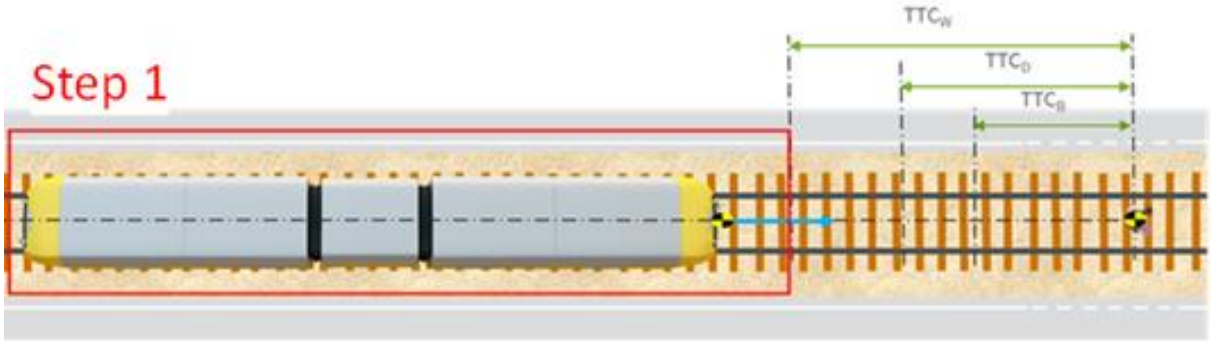


Figure 8. Vehicle level test case Step 1

Pass/Fail criteria:

- TCS_1 - Step 1 – Train
  - Expected result:
    - Warning = Not present
    - Deceleration = Not present
    - Braking = Not present
- TCS_1 – Step 1 – Sense
  - Expected result:
    - No object detected
- TCS_1 – Step 1 – Logic
  - Expected result:
    - No object detected (distance > TTC_W)
- TCS_1 – Step 1 – Actuator
  - Expected result:
    - No warning triggered
    - No deceleration actuated
    - No braking actuated

3.3.2.2.2. ***Step 2.*** **When the distance between the train and pedestrian is equal to the TTC_W, the intended functionality shall evaluate the detected object (pedestrian), calculating its distance and collision probability and warn the driver about the pedestrian on the train path.**

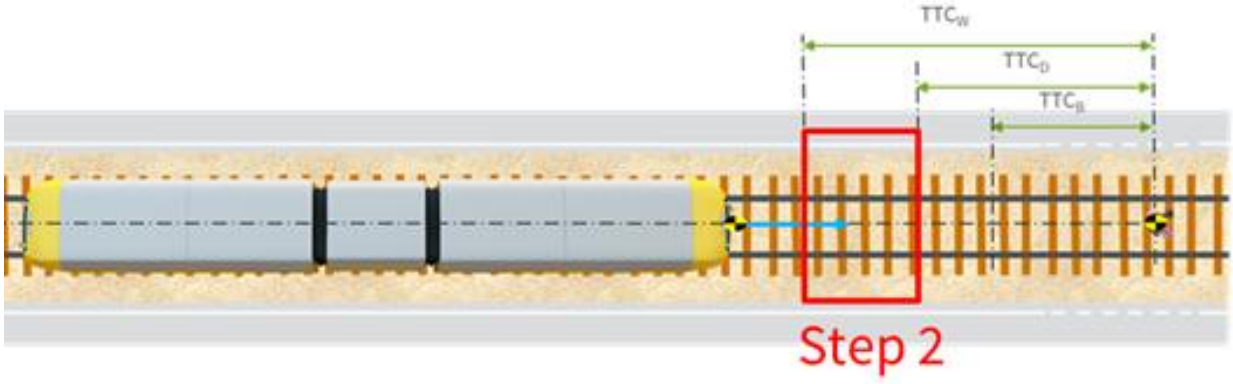


Figure 9. Vehicle level test case Step 2

Pass/Fail criteria:

- TCS_1 - Step 2 – Train
  - Expected result:
    - Warning = Present
    - Deceleration = Not present
    - Braking = Not present
- TCS_1 – Step 2 – Sense
  - Expected result:
    - Object detected
    - Object classified as “pedestrian”
      - Evaluate outputs of sensors to evaluate the expected results (e.g. detected objects, object classification)
- TCS_1 – Step 2 – Logic
  - Expected result:
    - Object evaluated as “collision” relevant (distance > TTC_D > TTC_B)
    - Evaluate outputs of Logic to evaluate the expected results (e.g. request to the actuator)
- TCS_1 – Step 2 – Actuator
  - Expected result:
    - Warning triggered
    - No deceleration actuated
    - No braking actuated

3.3.2.2.3. ***Step 3***. **When the distance between the train and pedestrian is equal to the TTC_D, the pedestrian is evaluated as critical and the intended functionality shall decelerate the train.**

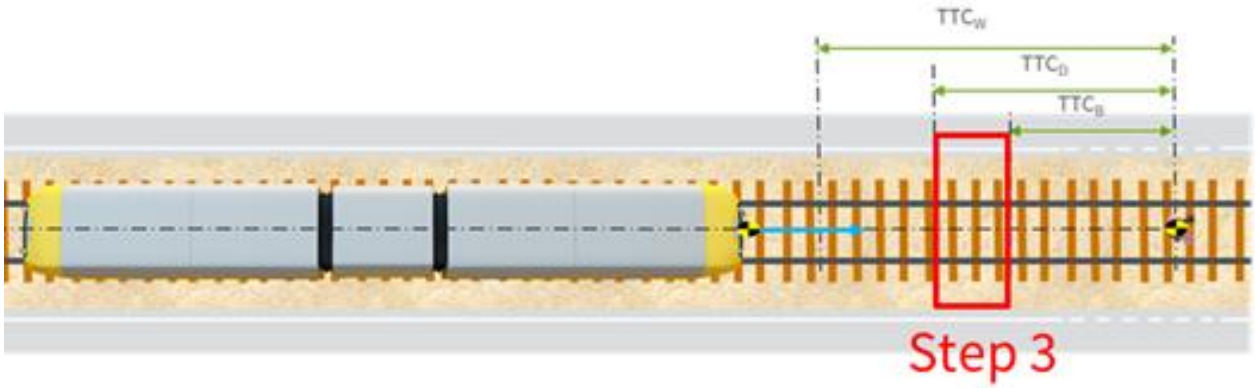


Figure 10. Vehicle level test case Step 3

Pass/Fail criteria:

- TCS_1 - Step 3 – Train
  - Expected result:
    - Warning = Present
    - Deceleration = Present
    - Braking = Not present
- TCS_1 – Step 3 – Sense
  - Expected result:
  - Object detected
  - Object classified as “pedestrian”
    - Evaluate outputs of sensors to evaluate the expected results (e.g. detected objects, object classification)
- TCS_1 – Step 3 – Logic
  - Expected result:
  - Object evaluated as “collision” relevant (distance = TTC_D)
    - Evaluate outputs of Logic to evaluate the expected results (e.g. request to the actuator)
- TCS_1 – Step 3 – Actuator
  - Expected result:
    - Warning triggered
    - Deceleration actuated
    - No braking actuated

3.3.2.2.4. ***Step 4.*** **When the distance between the train and pedestrian is equal to the TTC_B, the driver shall brake the train.**

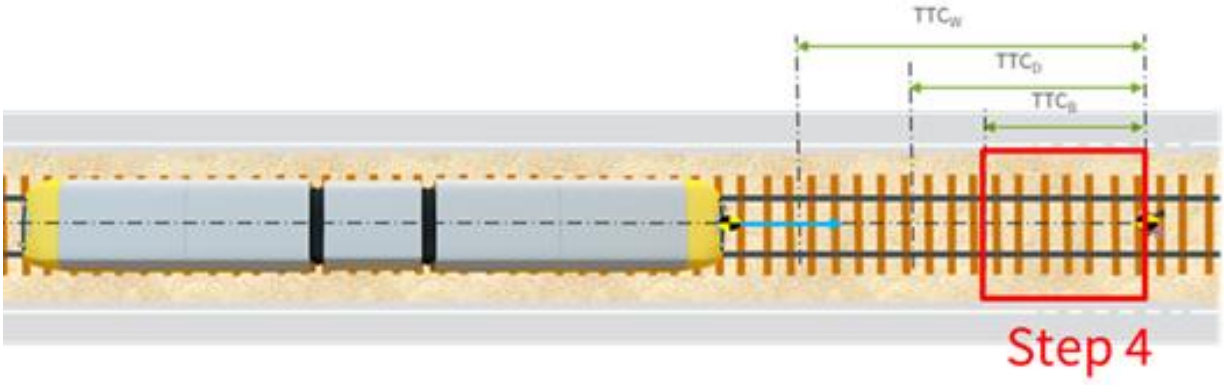


Figure 11. Vehicle level test case Step 4

Pass/Fail criteria:

- TCS_1 - Step 4 – Train
  - Expected result:
    - Warning = Present
    - Deceleration = Present
    - Braking = Present
- TCS_1 – Step 4 – Sense
  - Expected result:
    - Object detected
    - Object classified as “pedestrian”
      - Evaluate outputs of sensors to evaluate the expected results (e.g. detected objects, object classification)
- TCS_1 – Step 4 – Logic
  - Expected result:
    - Object evaluated as “collision” relevant (distance = TTC_B)
      - Evaluate outputs of Logic to evaluate the expected results (e.g. brake request to the driver)
- TCS_1 – Step 4 – Actuator
  - Expected result:
    - Warning triggered
    - Deceleration actuated
    - No braking actuated

3.3.2.3. *From Hazard Vehicle Level to Triggering condition*

- Hazard Vehicle Level:
  - The ATO fails to warn the driver when required.
- Control Action:
  - Warn the driver when a potential collision is imminent.
- Control Action Failure Mode:
  - Warning not provided in a critical scenario.
- Functional Insufficiency:
  - The ATO does not provide the warning.
- Triggering Condition:
  - Warning is not provided to HMI ECU due to low buses communication.
  - Missing object detection due to wrong camera position/camera calibration (e.g. The sensor is mounted in a wrong position/orientation leading to a missing object detection)
  - Missing object detection due to camera low performances (e.g. damaged lens or undetected offline sensor)
  - The logic responsible to elaborate the input data is stuck due to low computational power (e.g. low resources available).

3.3.2.4. *Example test case related to the triggering condition*

- Test ID: TCTC-01
- Tested Triggering Condition ID: TC3-001
- Test Description: The test aims to verify whether the ATO is deactivated against a failure in the camera.
- Test Steps:
  - ***Step 1:***
    - Initial state:
      - Kl.15 = on
      - No warning message available
      - Intended functionality state: active
    - Operating Element:
      - Camera Sensors
      - CAN Bus Simulator
      - ATO ECU
    - Operating:
      - Sent camera status signal to ATO ECU reporting that no failure is present
    - Expected Result:
      - No warning message provided
      - Intended functionality state: active
  - ***Step 2:***
    - Initial state:
      - Kl.15 = on
      - No warning message available
      - Intended functionality state: active
    - Operating Element:
      - Camera Sensors

        - CAN Bus Simulator
        - ATO ECU
    - Operating:
        - After x ms sent camera status signal to ATO ECU reporting that a failure is present
    - Expected Result:
        - Intervention of dedicated camera failure safety mechanism detecting the failure present in the camera causing the deactivation of the functionality within x ms.
    - Intended functionality state: deactivated

3.3.3. Aerospace Use Case

The following subsection focuses on implementing the V&V strategy in the aerospace use case.

3.3.3.1. Example of Scenario from the scenario catalogue

This subsection reports an example of one of the scenarios included in the scenario catalogue adapted to the aerospace use case developed by AIKO. It represents an Agent flies near a docking target and a black image is acquired due to a camera malfunction, as depicted in Figure 12.

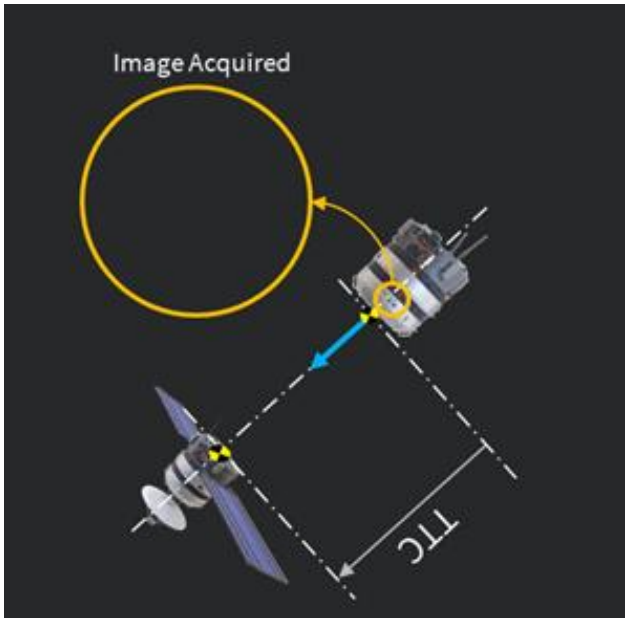


Figure 12. Visual representation of the scenario example.

Description of the expect behaviour:

- When the agent flies in the Earth Orbit (LEO/GEO) close to a docking target, the acquired image is black (null image) due to a camera malfunction.
- As soon as a camera malfunction, which can lead to an incorrect execution of the functionality, is detected, the intended functionality shall provide a warning to the driver, and the functionality shall be deactivated.

The scenario conditions/constraints are the following:

- The agent flies towards a docking target and is at a distance corresponding to a Time To Collision (TTC) of at least 200 s.
- The agent relative speed range is [0 cm/s, 10 cm/s]
- Initial distance from the docking target: 15 m
- Camera is not working, and null image is acquired
- The following environmental conditions shall be present:
  - Luminosity range: [1-30] (intensity of pixels in the image)
  - Microgravity: ag = 8,8 m/s2
  - LEO (Low Earth Orbit)
  - GEO (Geostationary Earth Orbit)
- The following Pre-conditions shall be respected:
  - Camera acquires null image (black image) due to a camera malfunction.

3.3.3.2. *Example of Vehicle level test case*

The following intended functionality capabilities shall be demonstrated:

3.3.3.2.1. ***Step 1.*** As the agent flies along its own path, it approaches the target.

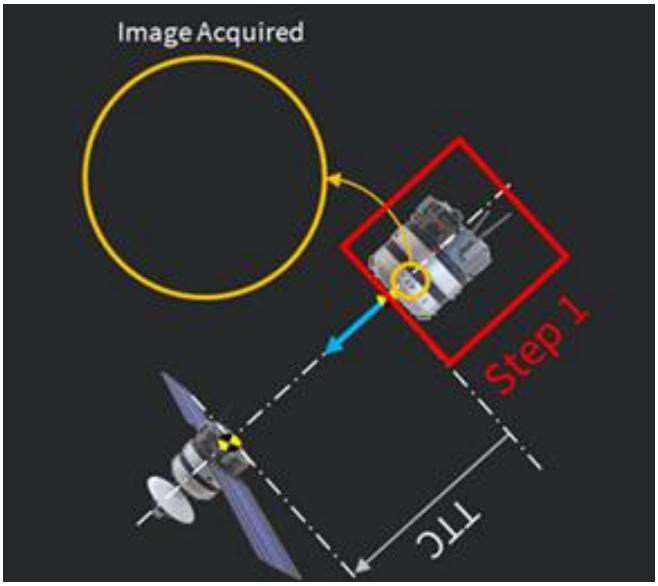


Figure 13. Vehicle level test case Step 1

Pass/Fail criteria:

- TCS_2 - Step 1 – Agent
  - Expected result:
    - Target: Not detected
    - Docking site: Not detected
    - Docking operation relevant information: Not provided
- TCS_2 – Step 1 – Sense
  - Expected result:
    - No object detected
- TCS_2 – Step 1 – Logic
  - Expected result:
    - No object detected (distance > TTC)
- TCS_2 – Step 1 – Human-Machine Interface (HMI)
  - Expected result:
    - Warning: Not triggered
    - Docking operation relevant information: Not provided

3.3.3.2.2. ***Step 2***. **When the distance between the agent and the target is equal to TTC, the intended functionality is not able to detect the target, since the acquired image is null (black) image due to a camera malfunction; the intended functionality shall provide a warning to the driver, and it shall be deactivated.**

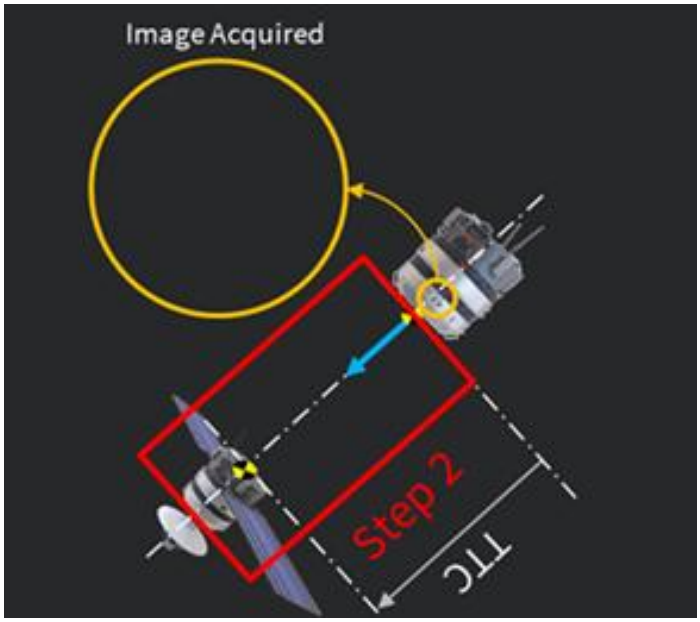


Figure 14. Vehicle level test case Step 2

Pass/Fail criteria:

- TCS_2 - Step 2 – Agent
  - Expected result:
    - Warning: Provided
- TCS_2 – Step 2 – Sense
  - Expected result:
    - No object detected
    - Null image acquired
- TCS_2 – Step 2 – Logic
  - Expected result:
    - Detection of malfunction in the camera
- TCS_2 – Step 2 – HMI
  - Expected result:
    - Warning: Provided

3.3.3.3. *From Hazard Vehicle Level to Triggering condition*

- Hazard Vehicle Level:
  - The GNC System wrongly provide information related to the docking operation.
- Control Action:
  - Information needed to docking operation are provided to the driver.
- Control Action Failure Mode:
  - Wrong information related to the docking operation are provided to the driver.
- Functional Insufficiency:
  - The GNC algorithm wrongly elaborate the received data.
- Triggering Condition:
  - The decision algorithm wrongly elaborates the received signal, due to unknown signal.
  - Missing detection of wrong input via plausibility check leads to incorrect agent position calculation.
  - Wrong agent parameter estimation due to wrong camera position/camera calibration (e.g. the sensor is mounted in a wrong position/orientation).
  - Agent parameters are wrongly estimated due to low computational power (e.g. overload).

3.3.3.4. *Example test case related to the triggering condition*

- Test ID: TCTC-01
- Tested Triggering Condition ID: TC1-001
- Test Description: The test aims to verify whether the GNC System is deactivated against a received signal with an unknown value.
- Test Steps:
  - ***Step 1:***
    - Initial state:
      - Kl.15 = on
      - No warning message available
      - Intended functionality state: active
    - Operating Element:
      - Camera Sensors
      - CAN Bus Simulator
      - GNC System ECU
    - Operating:
      - Sent camera signal to GNC System ECU with a valid value
    - Expected Result:
      - No warning message provided
      - Intended functionality state: active
  - ***Step 2:***
    - Initial state:
      - Kl.15 = on
      - No warning message available
      - Intended functionality state: active
    - Operating Element:
      - Camera Sensors
      - CAN Bus Simulator

- GNC System ECU
- Operating:
  - After x ms sent camera signal to GNC System ECU with unknown value
- Expected Result:
  - Intervention of dedicated CAN communication safety mechanism detecting the unknown value of camera signal causing the deactivation of the functionality within x ms.
  - Intended functionality state: deactivated

4. Acronyms and Abbreviations

Below is a list of acronyms and abbreviations employed in this document:

- AEB – Autonomous Emergency Braking
- AI – Artificial Intelligence
- ASPICE – Automotive SPICE
- DL – Deep Learning
- EASA – European Aviation Safety Agency
- Ego vehicle - vehicle fitted with functionality that is being analyzed for the SOTIF
- FSM – Functional Safety Management
- FuSa – Functional Safety
- GEO – Geostationary Earth Orbit
- HARA – Hazard Analysis and Risk Assessment
- II – Information Items
- ISO – International organization for standardization
- LEO – Low Earth Orbit
- ML – Machine Learning
- MLE – Machine Learning Engineering
- NN – Neural Network
- ODD – Operational Design Domain
- PAM – Process Assessment Model
- PRM – Process Reference Model
- QM – Quality Management
- QMS – Quality Management System
- SIL – Safety Integrity Level
- SOTIF – Safety Of the Intended Functionalities
- SPICE – Systems Process Improvement and Capability dEtermination
- SWE – Software Engineering
- TTC – Time To Collision
- TTW – Time To Warning
- VDA – Verband Der Automobil industrie
- V&V – Verification and Validation
- WP – Work Product

5. Bibliography

[1] ISO, ISO 26262:2018 - Road vehicles — Functional safety, 2018..

[2] ISO, ISO 21448:2022 Road vehicles — Safety of the intended functionality, 2022.

[3] VDA 702, «Situationskatalog E-Parameter nach ISO 26262-3,» 2015.

6. Annexes

This section collects the annexes attached together with the deliverable D2.3.

Annex A: Railway Safety Concept

This annex includes the document sent to the TÜV Rheinland certification entity to be reviewed.

Annex B: Review meeting presentation

This annex collects the power point shared with TÜV Rheinland during the assessment of the Railway Safety Concept presented in the Annex A. In that presentation, the main set of review comments received from the TÜV Rheinland entity are included.

Safe and Explainable
Critical Embedded Systems based on AI

Annex A: Railway Safety Concept

Safe and Explainable
Critical Embedded Systems based on AI

Railway Safety Concept

Version 1.0

Documentation Information

Contract Number	101069595
Project Website	www.safexplain.eu
Contratual Deadline	31.03.2025
Dissemination Level	PU
Nature	R
Author	Irune Yarza, Irune Agirre
Contributors	
Reviewer	Ralf Röhrig (TÜV Rheinland)
Keywords	Functional safety, AI, Railway

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

Change Log

Version	Description Change
V1.1	TÜV Rheinland corrections
V1.0	Final version for TÜV Rheinland
V0.1	First draft
V0.0	Document created

Table of Contents

Executive Summary

Deep Learning (DL) techniques are at the heart of most future advanced software functions in Critical Autonomous AI-based Systems (CAIS), where they also represent a major competitive factor. Hence, the economic success of CAIS industries (e.g., automotive, space, railway) depends on their ability to design, implement, qualify, and certify DL-based software products under bounded effort/cost. However, there is a fundamental gap between Functional Safety (FUSA) requirements on CAIS and the nature of DL solutions. This gap stems from the development process of DL libraries and affects high-level safety concepts such as (1) explainability and traceability, (2) suitability for varying safety requirements, (3) FUSA-compliant implementations, and (4) real-time constraints. As a matter of fact, the data-dependent and stochastic nature of DL algorithms clashes with current FUSA practice, which instead builds on deterministic, verifiable, and pass/fail test-based software.

The SAFEXPLAIN project tackles these challenges and targets by providing a flexible approach to allow the certification – hence adoption – of DL-based solutions in CAIS building on: (1) DL solutions that provide end-to-end traceability, with specific approaches to explain whether predictions can be trusted and strategies to reach (and prove) correct operation, in accordance to certification standards; (2) alternative and increasingly sophisticated design safety patterns for DL with varying criticality and fault tolerance requirements; (3) DL library implementations that adhere to safety requirements; and (4) computing platform configurations, to regain determinism, and probabilistic timing analyses, to handle the remaining nondeterminism.

This document provides strategies and specific solutions to realize safety patterns for safety-relevant software systems that include Artificial Intelligence (AI) software components. Moreover, the proposed architectural solution is tailored to a railway case study for its assessment with respect to safety certification by means of a safety concept.

Future exploitation of the main SAFEXPLAIN concepts, architecture and solutions could be at stake if no clear route for certification is identified, which can accompany industrial interest. Additionally, to ensure that relevant certification bodies with authority in the targeted domains are aware of the SAFEXPLAIN approach is also a key interest of the project.

This self-contained document is submitted to TÜV Rheinland, a relevant certification body in the industrial domain, with the following goals:

- The third-party authoritative review report on the suitability of the research concepts of SAFEXPLAIN, identifying possible implausibilities of the approach or possible contradictions with requirements of the Bases of Assessment and evaluating the use of the approach in the context mentioned.
- The dissemination of SAFEXPLAIN contribution to TÜV Rheinland.
- The gathering of detailed feedback from TÜV Rheinland.
- The definition of an action plan based on the received feedback (if needed).
- The suggestion of updates to the current standards or certification processes (where needed).

Structure of the document

This document is organized in three main parts:

- **Part I – Introduction (informative)**: This part serves solely introductory purposes and hence, it is not intended for review. Section 1 provides a basic background on the SAFEXPLAIN project and certification standards, focusing on the main topic of the project: Artificial Intelligence (AI) based systems, NVIDIA Jetson AGX Orin (target HW platform) as well as standards and technical reports.
- **Part II – System architecture (to be reviewed):** This part describes the reference safety architecture and safety patterns to be reviewed by TÜV Rheinland:
  - Section 2 reviews the relevant sources of risk for AI-based solutions in safety-critical systems, including AI-related factors and traditional ones. Then, section 2.2 reviews the different usage and automation levels for DL solutions, and how they are tackled incrementally in the project, and also in this document.
  - Section 3 provides the general structure of the safety patterns, as well as the techniques considered for diverse redundancy, diagnostics, monitoring, and supervision in the different safety patters.
- **Part III – Railway case study (to be reviewed):** This part tailors the reference safety architectural pattern to a railway case study.
  - Section 4 presents the railway case study, which consists of a railway domain object detector controller for an Automatic Train Operation (ATO) system. Aiming an incremental adoption of AI in the system, several situations have been considered, where the AI system plays different roles within the ATO system.
  - Section 5 instantiates the general pattern for specific usage and automation levels, reviews the relevant techniques to be used for diverse redundancy, diagnostics, monitoring, and supervision for each of the patterns, and provides guidance on how to map those solutions to the target platform of the project (NVIDIA Orin).

Part I – Introduction (informative for TÜV Rheinland)

1. Introduction

Systems with safety requirements follow a development process that leads to architectures that allow satisfying those requirements. Those architectures have a large set of commonalities across different systems with comparable safety requirements, and hence, a number of safety patterns have been developed in each application domain as a way to generate specific architectures reusing previous efforts to reduce costs and risks. One such example of safety pattern is the E-gas Concept (EGAS Workgroup, 2013) used in the automotive domain, which is the basis for the architecture of many automotive systems with limited integrity levels (e.g., ASIL-A or ASIL-B according to ISO 26262 (International Standardization Organization, 2018)).

However, AI-based software has a number of characteristics that clash with those expected for safety-relevant software, such as a pseudo stochastic behaviour with non-guaranteed correct outputs, and a data-based software design rather than being a purely control algorithm. Hence, traditional safety patterns cannot be applied as they are, and new ones need to be devised.

In this section, we provide some context and scope to the problem of developing AI-based software architectures with safety requirements and provide some background on the relevant safety regulations and the target platform where those software architectures have to be deployed.

1.1. Background

This section provides background on the main concepts used in this report: nomenclature on AI-based systems, the main features of NVIDIA Jetson AGX Orin platform where the safety patterns will be applied, and a recall of the safety standards used as reference.

1.1.1. AI-based systems

When referring to AI-based safety systems, this report considers the definitions of the European Aviation Safety Agency (EASA) concept paper for Machine Learning (ML) application [1], which makes the decomposition shown in Figure 1.

Based on this decomposition, the EASA concept paper makes the following definitions:

- AI-based system: systems encompassing traditional subsystem(s) and incorporating at least one AI-based subsystem.
- AI-based subsystem: subsystem that involves one or more AI/ML constituents.
- AI/ML constituent: It is a combination of software and hardware items that include at least one specialized hardware or software item containing at least one ML model.
- AI/ML item: specialized hardware or software item that builds the ML model(s). In particular, we focus on Deep Learning (DL) models, a subfield of ML.
- Traditional subsystem: subsystem that does not include any ML model.
- Traditional SW/HW item: hardware or software items that do not include ML model(s).

This safety concept focuses on AI/ML constituents with the following features:

- Deep Learning (DL) algorithms based on supervised learning for visual perception classification tasks.
- Applications based on offline learning processes in which the model remains fixed at approval time, while excluding online learning processes.

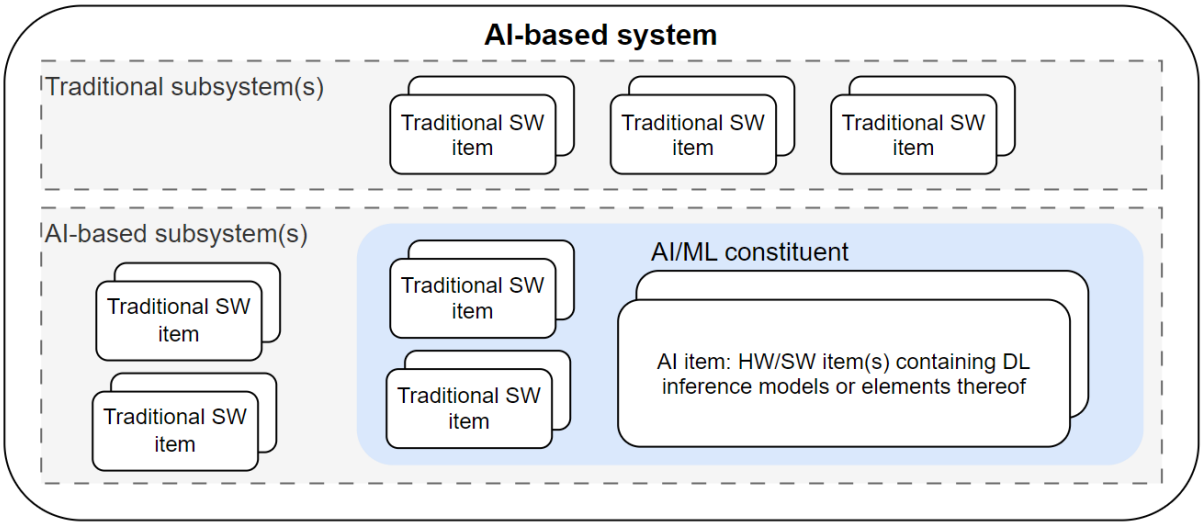


Figure 1: Artificial Intelligence (AI)-based system decomposition based on EASA concept paper [1]

1.1.2. NVIDIA Jetson AGX Orin

The target MPSoC platform in SAFEXPLAIN belongs to the NVIDIA Jetson AGX Orin family. In SAFEXPLAIN, the AGX Orin Dev Kit has been selected. The Orin comprises 3 clusters of 4 Arm Cortex-A78AE CPUs, a dual core Sensor Processing Engine (SPE), which can be configured in lockstep mode, a NVIDIA Ampere GPU, ad-hoc AI-oriented accelerators, and a video encoder / decoder.

Clusters based on Arm cores provide general purpose computing capabilities suitable for control software and software components around AI-based software, but without needing AI-specific support or massively parallel hardware components. Each of the three clusters includes four cores that can be used as either 4 independent cores, or as 2 pairs of lockstep cores. Cores include local (L1 and L2) caches. There is a L3 cache shared across all cores in the cluster, but it includes partitioning support. We refer to the set of 3 core clusters as CCPLEX for short.

Computing devices like the GPU and the AI-oriented accelerators have a complementary nature to the core clusters and are suitable for AI software and other embarrassingly parallel software. In particular, the GPU has a number of Stream Multiprocessors (SMs) sharing a L2 cache, which, apparently, cannot be partitioned.

Core clusters and the GPU share a L4 cache that, so far, cannot be partitioned, and hence, both cores and the GPU share it. However, we note that, due to its limited size, does not seem to be intended for providing increased cache space for the L3 caches of the core clusters or the L2 cache of the GPU.

1.1.3. Standards and technical reports

Although this safety concept is focused on a railway domain case study, the outcomes of this project are meant to be cross-domains, therefore, the following domain independent standards are taken as basis:

- **IEC 61508:2010 - Functional safety of electrical/electronic/programmable electronic safety-related systems** [2]**:** Traditional FUSA architectural safety patterns from this standard are used as starting point for the definition of reference architecture of Section 3 in this report.

- **ISO/IEC TR 5469:2024 - Artificial intelligence — Functional safety and AI systems** [3]**:** Section 2.2 explains the definition of the different DL usage levels according to ISO/IEC TR 5469 which is used as the basis for the incremental strategy of this report and the architectural safety patterns for AI presented in Section 5.

In addition to these domain independent standards, technical reports from different domains are also taken as reference:

- **CAST-32A and AMC-20-193 from the avionics domain for multicore processors** [4]: these two technical reports from the avionics domain provide guidance on the adoption of multicore processors for critical systems. Many of the principles described there are also applicable to HPEC platforms that integrate multicore processors together with additional accelerators and diverse computing resources.
- **E-Gas architecture concept from the automotive domain** [5]: The E-Gas architecture concept, defined by the German Association of the Automotive Industry (VDA), has the aim to standardize the safety architecture for engine control systems. The defined architecture is in compliance with ISO 26262 and it can be applied as a well-trusted design principle. The E-Gas defines a 3-level monitoring concept that has been used as reference in this report (Section 3.2), adapting it to AI systems particularities and to ISO/IEC TR 5469.

Part II– Reference safety architecture

2. Sources of Risk and Incremental Strategy

In order to define the safety architecture patterns, first main problems arising from the use of AI in safety critical systems and their root causes are analysed and then an incremental strategy to deal with them is defined.

2.1. AI Risk factors

As for traditional safety related systems, the main risk factors related to AI-based systems can be originated either in its development or at operation. Figure 2 depicts this classification criteria together with information sources used in this section to identify the different risk factors. During the development, there are several stages at which systematic faults can originate, like incomplete specifications, biased training data or wrong design decisions affecting the ML model. In order to reduce the probability of systematic faults in the AI-based subsystem, safety considerations shall be taken in the AI lifecycle. In SAFEXPLAIN this has been addressed by the AI-FSM [6] and hence, this report does not focus on the risk factors originated in the AI lifecycle but on the AI Operation instead. However, the high complexity of AI systems makes it very difficult to mitigate all systematic faults through design and verification and validation processes, and it is assumed that residual systematic faults will manifest at operation and should be covered by the runtime mechanisms in the safety architecture too, as done for random faults.

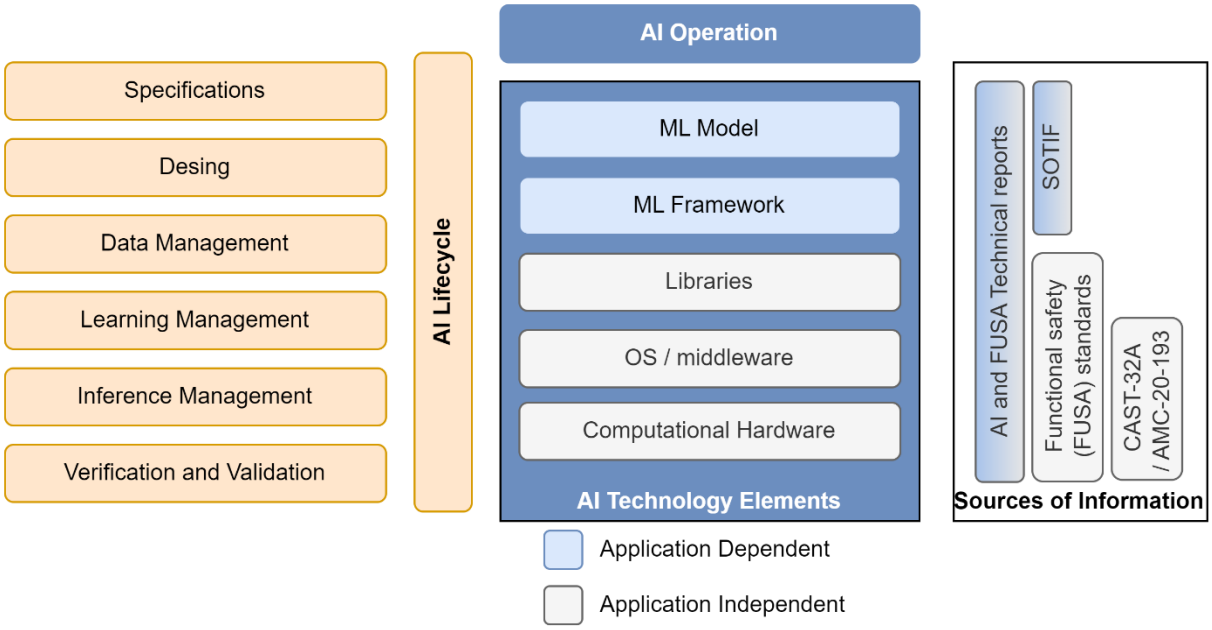


Figure 2: AI Risk factor classification and sources of information

At operation time, residual systematic faults and random faults can impact different elements of the AI-subsystem, this determines the architectural measures required for runtime error detection and monitoring. Based on ISO/IEC TR 5469 we classify these AI technology elements in the 5 groups shown in Figure 2:

- Application Independent: The lowest levels of the AI technology elements can be considered as application independent:
  - Computational Hardware: refers to the hardware platform and its processing units (e.g., CPU, GPU, Accelerators).

- OS / middleware: This component is not included in ISO/IEC TR 5469, but many faults could also originate on this layer.
- Libraries: ML components often require specific libraries for their inference (e.g., cuDNN). These libraries can also be implemented in different programming languages such as CUDA, C, C++, python…

- Application dependent: Higher levels of the AI technology elements are usually platform dependent:
  - ML framework: many different frameworks can be used depending on the application that is being implemented (e.g., TensorFlow, Pytorch, Keras).
  - ML model: the ML model will depend on the specific application and the design decisions taken for it (architecture, hyperparameters, number of layers, etc).

The analysed sources for identifying relevant risk factors on AI operation cover different AI technology elements as explained in next subsections.

2.1.1. Functional Safety Standards

Functional safety standards such as IEC 61508, ISO 26262 or EN 5012x only address the application independent technology elements from Figure 2 (computational hardware and software) as they do not address AI-specific topics. According to ISO / IEC TR 5469 these technology elements can be addressed through existing functional safety concepts following IEC 61508-2 for hardware and IEC 61508-3 for software. However, for elements containing AI technology (ML model and related tools) a set of new properties, listed in Section 2.1.3, are defined.

Traditional functional safety standards classify failures based on their origin as systematic or random:

- **Systematic failure**: “failure, related in a deterministic way to a certain cause, which can only be eliminated by a modification of the design or of the manufacturing process, operational procedures, documentation or other relevant factors” from IEC 61508-4.
- **Random hardware failure**: “failure, occurring at a random time, which results from one or more of the possible degradation mechanisms in the hardware” from IEC 61508-4.

Concerning systematic faults affecting AI SW development (lifecycle) it is an open research topic that has been explored by several research papers. Humbatova et al. [7] introduce a large taxonomy of faults in DL systems’ development using several frameworks. Moreover, Schnitzer et al. [8] propose a framework for the systematic management of risks associated with AI. This framework builds upon an AI hazard list from a SoA analysis.

2.1.2. Technical reports on multicore integration (CAST-32A / AMC-20-193)

CAST-32A / AMC-20-193 focus on multicore processors and therefore mostly on the hardware computing element and allocation of resources to deal with interferences. The following main risk factors are considered:

- Contention for resources and interference between software applications or tasks even if there is no explicit data sharing among concurrent tasks, low level resources are shared (cache or interconnects) coupling exists on the platform level, which can cause interference between them.

- Interference caused by the arbitration of shared resources.
- Verification of the use of shared resources: demonstrate that the hosted software applications function correctly and have sufficient time to execute in the presence of the interference that occurs when all the hosted software is executing on a multicore. The WCET of a software component or task may increase significantly when other software components or tasks are executing in parallel on the other cores.
- Exceedance of resource capabilities by software applications.
- Hardware dynamic features that can alter system behaviour (energy saving features, clock enable / gating, frequency adaptations, deactivating one or more cores, or dynamic control of peripheral access).
- Configuration settings: active cores, execution frequencies, priorities and allocation of shared resources (memory, cache, interconnect).

2.1.3. AI and FUSA Technical Reports (ISO/IEC TR 5469)

As previously stated, ISO/IEC TR 5469 refers to traditional functional safety standards for application independent AI technology elements, and defines the following 6 properties for AI application dependent ones:

- Degree of automation and control
- Degree of decision transparency and explainability
- Environmental complexity and vagueness of specifications
  - Operational Design Domain (ODD) complexity
  - ODD specification
  - Environmental changes: Data drift (training data does not match runtime domain), Concept drift (statistical properties of data change over time)
- Adversarial inputs
- AI Hardware issues
- Technological maturity

2.1.4. Safety of The Intended Functionality (SOTIF) (ISO/DIS 21448)

With the increase of advanced functionalities and automation, ISO/DIS 21448 acknowledges that many hazards are not covered by functional safety standards and defines the concept of functional insufficiencies, which can happen even when the system is free from systematic and random faults addressed in functional safety standards. Functional insufficiencies, which could be considered as a type of systematic faults, are defined as one of the following:

- Insufficiency of specification: This mainly affects to the AI lifecycle as it is defined as “specification, possibly incomplete, contributing to either a hazardous behaviour or an inability to prevent or detect and mitigate a reasonably foreseeable indirect misuse”.
- Performance insufficiency: “limitation of the technical capability contributing to a hazardous behaviour or inability to prevent or detect and mitigate reasonably foreseeable indirect misuse”.

2.1.5. Summary of risk factors

Table 1 summarises the risk factors described within Section 2.1 mapping them to the affected AI-technology elements the source of information.

Table 1: Summary of risk factors

Risk factor	AI technology element	Source
Traditional FUSA risk factors (systematic / random)	All AI technology elements (i.e., computational hardware, OS / middleware, libraries, ML framework and ML model).	IEC 61508 and derived FUSA standards (ISO 26262, EN 5012x, …)
HPEC platform integration risk factors	Computational hardware and lower-level SW (i.e., OS / middleware).	CAST-32A / AMC-20-193 ISO 26262 part 11
AI performance insufficiency	ML model or framework.	ISO/DIS 21448 (SOTIF)
AI & FUSA risk factors (low/medium integrity level)	All AI technology elements (i.e., computational hardware, OS / middleware, libraries, ML framework and ML model).	AI and FUSA Technical Reports (ISO/IEC TR 5469)

2.2. Incremental strategy

ISO / IEC TR 5469 describes different DL usage levels:

- **DL usage Level A1**: AI technology is used and it is possible to make automated decision of the system function using AI technology.
- **DL usage Level A2**: 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).
- **DL usage Level C**: AI technology is not part of a safety function but can have an impact on it.
- **DL usage Level D**: AI technology is not part of a safety function and does not have an impact on it due to sufficient segregation and behaviour control.

Similarly, the EASA concept paper classifies AI systems based on the level of automation / assistance to the user:

- **EASA Level 1**: AI used for assistance to human (human augmentation or cognitive assistance in decision and action selection).
- **EASA Level 2**: Human-machine teaming (cooperation or collaboration).
- **EASA Level 3**: Autonomous AI-based decisions and actions.

Based on these classifications, in SAFEXPLAIN we have defined the incremental strategy shown in Figure 3 for AI adoption in safety critical systems, which incrementally addresses the risk factors of Section 2.1. This strategi defines first a reference safety architecture (described in Section 3), which is tailored to the railway case-study and instantiated to three safety patterns in order to address the different DL usage levels:

- Safety Pattern 1 (SP1) (described in Section 5.1) aims to address up to a DL usage level D or EASA Level 1. The AI/ML constituent is not part of the safety function, therefore, we set the focus on safety techniques and measures to detect and mitigate errors associated with traditional FUSA and HPEC platform integration risk factors.
- Safety Pattern 2 (SP2) (described in Section 5.2) aims to address up to a DL usage level C or EASA Level 2. The AI/ML constituent is not part of the safety function although it collaborates on the decision and can therefore have an impact on it, so we assume that the AI/ML

constituent has a low/medium SIL. This SP considers the techniques and measures of SP1 and sets the focus on solutions to detect and mitigate errors associated with AI performance insufficiencies as well as AI and FUSA risk factors on the ML framework and DL model.

- Safety Pattern 3 (SP3) (described in Section 5.3) aims to address up to a DL usage level A1 or EASA Level 3, where the AI/ML constituent is part of the safety function. The AI/ML constituent provides autonomous AI-based decisions and actions, so it has a high SIL. SP3 considers the techniques and measures from previous SPs and sets the focus on achieving higher integrity level.

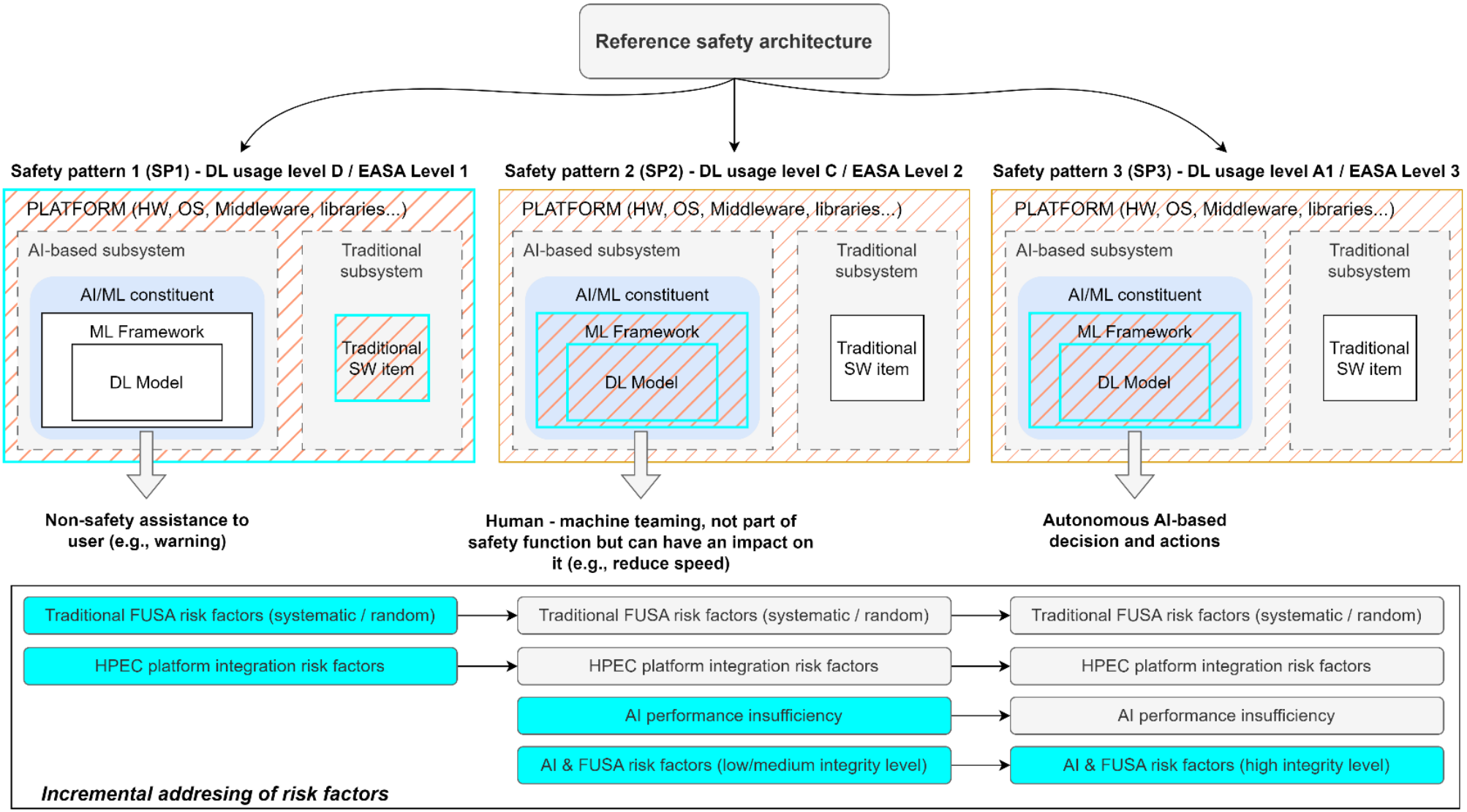


Figure 3: Incremental strategy for AI adoption in safety critical systems

3. Reference architecture pattern

Figure 4 depicts a reference safety architecture pattern and its main safety elements, illustrated by means of enumerated rhombus symbols. These elements constitute the main building blocks to later define the specific safety architectural patterns for each DL usage level.

The goal of the reference safety architecture pattern is to show the main elements proposed in this report and an example on how they can be integrated together to build a safety critical system. This reference architecture shall be later tailored and adapted to each use case, as it is done in Section 5 with the adoption of the safety patterns to the railway case-study. Hence, the reference architecture is meant to be used as a baseline to define different variants according to particular needs on case-studies.

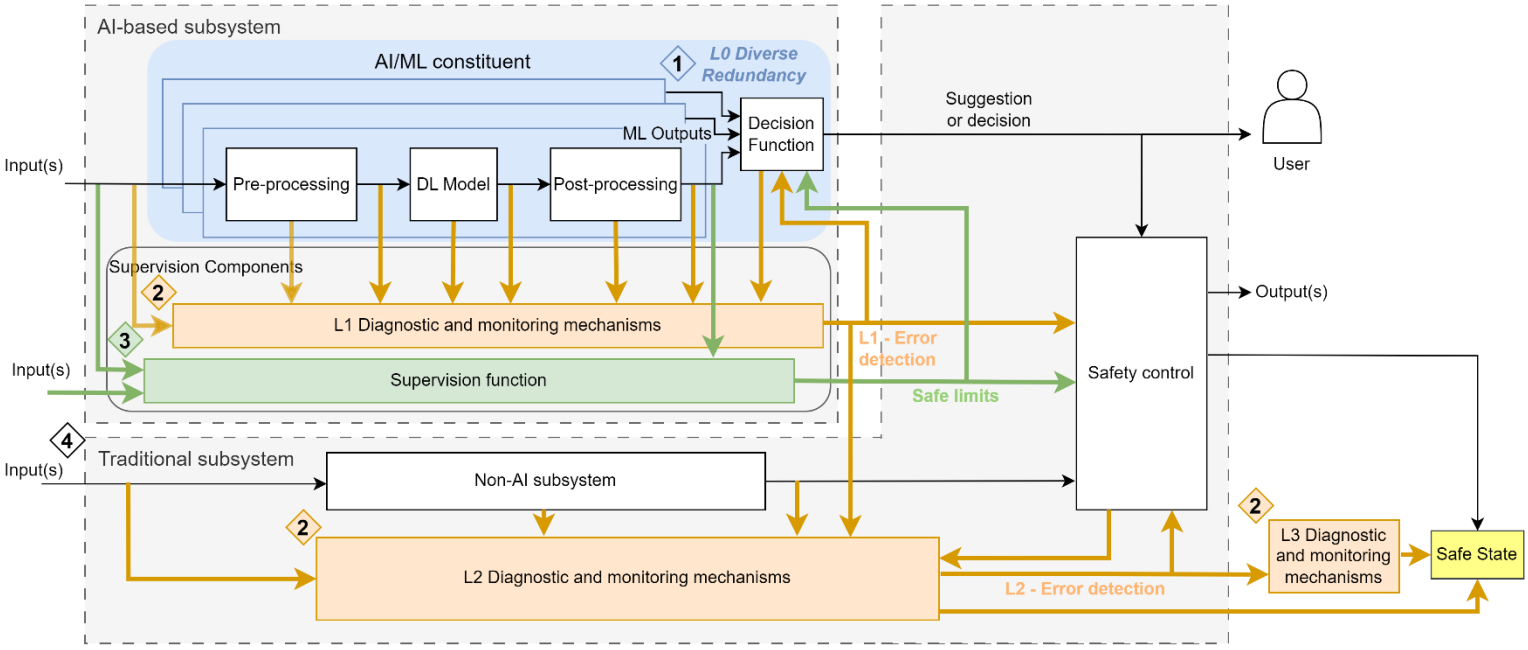


Figure 4: Reference safety architecture pattern for safe AI-based systems

In summary, the reference architecture pattern of Figure 4 integrates the following four safety mechanisms on it:

1. Diverse redundancy within the AI/ML constituent (Subsection 3.1). Note that redundancy and diversity are commonly also applied to traditional systems, however we consider such part out of the scope of this document, and we just focus on its application to the DL component.
2. Diagnostic and monitoring mechanisms (Subsection 3.2). A hierarchical diagnostic and monitoring concept consisting of three levels is defined.
3. Supervision function (Subsection 3.3). Its main goal is to check the appropriateness of the environment and to supervise the output of the ML constituent to identify unsafe situations and stablish the limits for safe operation, providing a safe envelope.

All in all, the reference safety architecture pattern shown in Figure 4, consists of two main subsystems:

- **AI-based subsystem**. Comprises all the AI-related elements in the architecture.
  - **AI/ML constituent**. Consists of the ML model as well as the corresponding data pre-processing and post-processing components.

- **Decision Function**. The AI/ML constituent implements a diverse redundancy scheme (see Section 3.1) to enhance the performance of the DL model and increase the ration of AI subsystem errors that can be detected and controlled. Within this diverse redundancy scheme, the Decision Function, takes the output from all the redundant nodes and provides a single prediction. Then, according to the DL usage level, the Decision Function might provide a suggestion (e.g., warning message, status information) and / or decision (e.g., command on the actuators) to the user and / or safety control.
- **Supervision components.** Consists of the L1 diagnostic and monitoring (L1DM) mechanisms as well as the supervision function.

  The former, comprises the lower-level diagnostics and monitoring in the platform, which complements the DL model diverse redundancy for detecting runtime errors or model insufficiencies and anomalies on the AI/ML constituent (see Section 3.2.1). To this end, the L1DM mechanisms gathers information from the AI/ML constituent (i.e., inputs, pre-processing, DL model, post-processing, decision function) and provides information on the detected errors to the decision function and L2 diagnostics and monitoring (L2DM) mechanism.

  The latter, bounds the AI/ML constituent operations to work within a predefined safety envelope (see Section 3.3). To this end, it collects information from the inputs/outputs of the AI/ML constituent and provides a set of constraints or limits to the decision function.

- **Traditional subsystem**. Comprises the components usually present in a traditional (non-AI based) system (see Section 3.4).
  - **Non-AI subsystem**. Depending on the DL usage level, the non-AI subsystem can have different usage, it can either be the main safety element, or a fallback element that is activated whenever the supervisor identifies an unsafe operation of the AI-based subsystem.
  - **L2 diagnostic and monitoring mechanisms.** Comprises platform-level diagnostic and monitoring following traditional functional safety techniques (see Section 3.2.2). To this end, it gathers information from all the components within the traditional subsystem (i.e., inputs, non-AI subsystem, safety control) as well as the L1DM mechanism (i.e., L1 – Error detection) and provides information on the detected errors (L2 – Error detection) to the safety control and L3 diagnostics and monitoring (L3DM) mechanism. Considering the error information from lower-level diagnostic and monitoring mechanisms (i.e., L0 and L1), the L2DM is responsible of triggering the required reaction for error handling (e.g., take the system to a safe state).
  - **Safety control**. Collects the output information from the AI/ML constituent and the non-AI subsystem as well as safety related information from the diagnostic and monitoring components (i.e., L1DM, L2DM) and the supervision function (i.e., safe limits) to implement the system control and actuation logic (i.e., command system output(s) or take the system to a safe state). For instance, if the non-AI subsystem implements a fallback function for the ML constituent, in the event that a functional safety problem is detected in the AI-subsystem, the safety control might decide to discard the suggestion/decision from the decision function and operate according to the fallback

function. The safety control might also limit the output(s) to meet with the safe limits defined in the supervision function or even take the system to a safe state.

- **L3 Diagnostic and monitoring mechanisms (L3DM)**. Following traditional functional safety practices some diagnostic may be implemented on an external device such as an external watchdog or microcontroller. Considering the error information from the L2DM mechanisms (i.e., L2 – Error detection), the L3DM mechanism is responsible of triggering the required reaction for error handling (e.g., take the system to a safe state).

Elements with no colouring in the reference safety architecture pattern (Figure 4) are out of the scope of the project, since they are not directly related to safety techniques and measures for the AI execution and may vary depending on the system application area.

3.1. Diverse redundancy

The goal of redundancy and diversity on the DL model is to enhance the performance and to increase the portion of AI subsystem errors that can be detected and controlled. In addition, redundancy can also improve system availability, by maintaining system operation even when one instance of the redundant architecture is failing. Redundancy shall be complemented with diversity in order to deal with systematic faults and model insufficiencies. When used for improved diagnostics, diverse redundancy can be considered as a particular technique of the AI subsystem diagnostic mechanisms of Subsection 3.2.1.

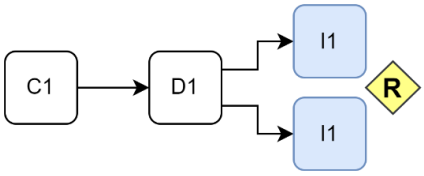
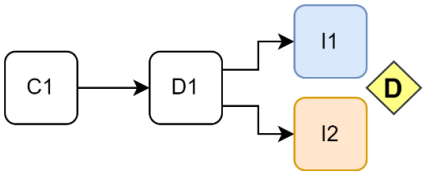
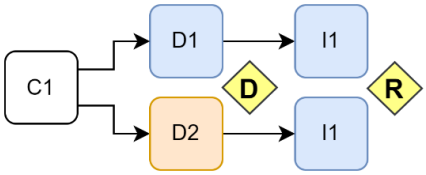
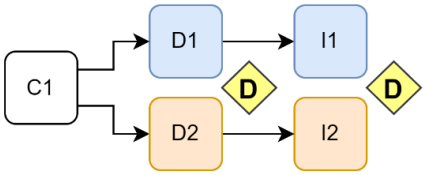
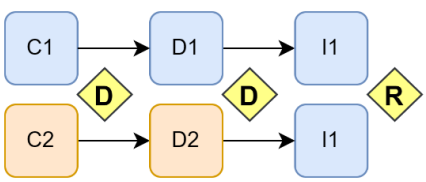
We base on the following three types of diverse redundancy that can be combined to build the Diverse Redundancy Schemes (DRS) proposed in Table 2 with varying degrees of diagnostic coverage:

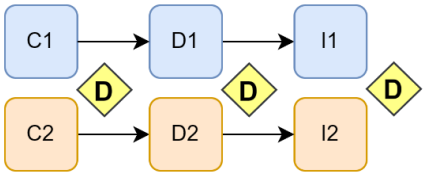
- Inference Platform diversity (identified as “I” in Table 2) – different platform resources for running the inference. Inference Platform diversity can be applied to different platform elements. These are some example approaches:
  - Inputs (e.g., diverse cameras, sensors, input image flips…)
  - Processing resources (accelerators, CPUs…)
  - Temporal (e.g., different time instants)
  - Data precision (e.g., float vs double, short vs int vs long int)
  - Data rounding (e.g., for floating point data)
  - Sources for potential random parameters
- DL model Development diversity (identified as “D” in Table 2): different techniques or elements for developing the DL model. DL model development diversity can be applied to different development phases and elements. These are some example approaches:
  - Model Architecture
  - Execution framework (e.g., TF lite, pytorch, darknet…)
  - Hyperparameters
  - Prediction criteria (e.g., confidence threshold, bounding box overlapping threshold, single vs multi class prediction…)
  - Training datasets
  - Training process (including convergence thresholds, weighting predictions…)
  - Training platform

- Concept diversity (identified as "C" in Table 2): different problem formulation with same final goal. The problem formulation may vary depending on the application are. These are some example approaches:
  - Object detection vs object part detection
  - Object detection vs obstacle free path detection

More specific application examples of each DRS are later proposed for each of the safety patterns (refer to Sections5.1, 5.2, and5.3).

Table 2: Diverse Redundancy Schemes

ID	Type of Diverse redundancy: Concept – Development – Inference	Diagnostic Coverage	Description
DRS_1		Low	Single DL model development, replicated in the inference platform without diversity (each replica uses same SW stack and processing elements (e.g., GPU)). Allows addressing traditional FUSA risk factors.
DRS_2		Low to medium	Single DL model development, replicated in the inference platform with diversity (each replica uses different resources (different inputs, processing elements (e.g., GPU and CPU), etc.)). Allows addressing traditional FUSA risk factors.
DRS_3		Low to medium	Development of two different DL models based on same concept (e.g., object detection) with diversity in the model development (training process, training data, model architecture, AI framework, etc.). Inference without diversity (each replica uses same SW stack and processing elements (e.g., GPU)). Allows addressing AI & traditional FUSA risk factors as well as AI performance insufficiencies.
DRS_4		Medium to high	Combination of DRS_2 and DRS_3 with DL model Development diversity and Inference Platform diversity. Allows addressing AI & traditional FUSA risk factors as well as AI performance insufficiencies.
DRS_5		Medium to high	Development of two different DL models based on different concepts (e.g., one for object detection and the other for obstacle free path detection). This requires DL model Development diversity. Inference without diversity (each replica uses same SW stack and processing elements (e.g., GPU)). Allows addressing AI & traditional FUSA risk factors as well as AI performance insufficiencies.

ID	Type of Diverse redundancy: Concept – Development – Inference	Diagnostic Coverage	Description
DRS_6		High	Combination of DRS_4 and DRS_5 with concept diversity, DL model Development diversity and Inference Platform diversity. Allows addressing AI & traditional FUSA risk factors as well as AI performance insufficiencies.

R Redundancy D Diverse Redundancy

Note that, differently to usual control algorithms where diverse redundancy is applied preserving bit-level precision (e.g., dual-core lockstep), some diversity schemes lead naturally only to semantic-level precision, meaning that we expect the same result in semantic terms[1] (e.g., same object detection in the same location), but results are very likely to be different at bit level (e.g., different confidence values, not fully identical bounding boxes, etc.). Therefore, mechanisms to produce the final (combined) prediction or report an error are needed, such as voting, averaging, and non-maximum suppression (NMS).

3.2. Diagnostic and Monitoring mechanisms

The reference safety architecture pattern is based on the hierarchical diagnostics and monitoring approach of Figure 5. This strategy seeks to decouple AI specific techniques from traditional functional safety approaches and to ease, in this way, the later tailoring to incremental safety patterns.

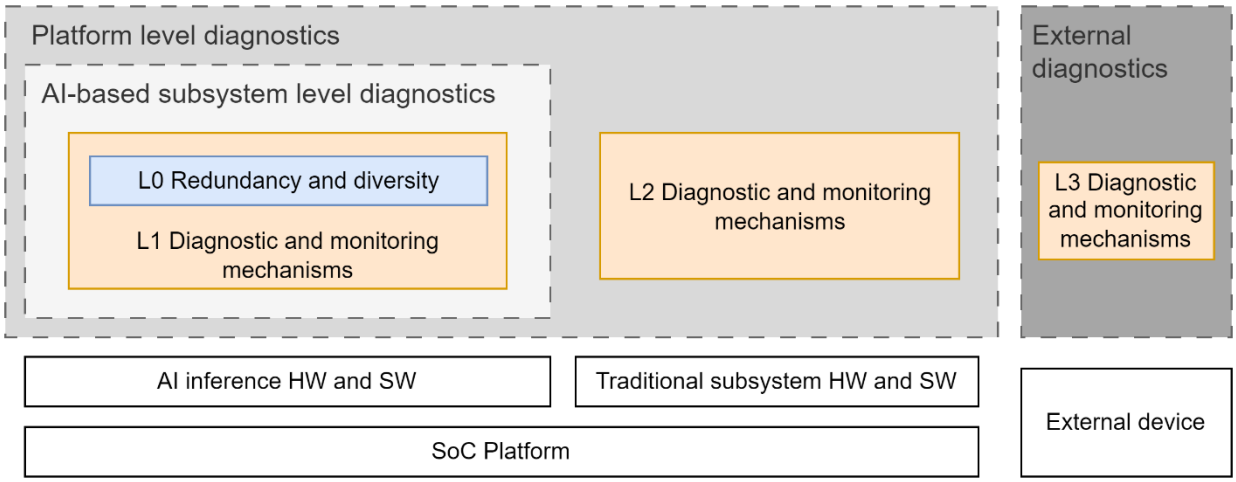


Figure 5: Hierarchical diagnostics and monitoring approach

[1] Two results are regarded as identical in semantic terms when, despite not being identical at bit level, both of them can be considered correct, and the impact of using one or another at system level is negligible or, simply, undistinguishable.

The proposed approach is based on four diagnostic levels (L0 to L3) to incrementally cover the errors on the following subsystems:

1. AI-based subsystem level diagnostics (L0 – L1): The redundancy and diversity (L0) described in previous subsection is complemented by additional diagnostic and monitoring mechanisms (L1) with the purpose of detecting runtime errors or model insufficiencies and anomalies on the AI subsystem and the elements required for its execution (e.g., accelerators, AI frameworks, etc.). Note that L1 diagnostics and monitoring mechanisms may, in turn, include AI components (e.g., to assess suitability of the data used for inference, or monitoring abnormal results from specific DNN layers).
2. Platform level diagnostics (L2): This includes L0 – L1 diagnostics and additional L2 diagnostics to detect runtime errors on additional platform HW and SW components following traditional functional safety practices and diagnostics techniques (e.g., memory self-tests, freedom from interference at platform level…).
3. External diagnostics (L3): Following traditional functional safety practices some diagnostic may be implemented on an external device such as an external watchdog or microcontroller.

Each of these diagnostic and monitoring levels is further described in next subsections, except L0 that has been already described in Subsection 3.1.

3.2.1. L1DM mechanisms – AI subsystem

As stated above, L1 diagnostic and monitoring mechanisms aim to diagnose the AI subsystem, and the platform resources required for its execution. All in all, the L1 mechanisms have the following purpose:

- Diagnose the AI model to detect runtime errors or model insufficiencies and anomalies.
- Diagnose the elements that participate in AI inference: complex HW and SW stack, black / grey box diagnostics.
- Monitor the use of resources at the AI subsystem level to guarantee freedom from interference at platform level (L2)

These mechanisms can be applied on different elements of the AI-subsystem. Bellow we present different L1 diagnostic mechanisms that may be applied to the different AI-subsystem elements.

- **Inputs:** diagnostic mechanisms for input correctness, data quality, data redundancy, temporal consistency…
  - Anomaly detectors
  - Input image comparison (e.g., consecutive input frames, input from redundant cameras).
- **Model:** diagnostic and monitoring mechanisms for execution errors, timing, program sequence, neuron activation patterns…
  - Generation of execution signatures.
- **Outputs:** diagnostic mechanisms for outputs, plausibility checks, input-output correlation, temporal consistency…
  - Output trajectory prediction
- **Resource usage:** monitoring mechanisms for resource usage (e.g., CPU/GPU usage, memory usage)

The following subsections provide further details on the implementation of the techniques for L1 diagnostic and monitoring.

3.2.2. L2DM mechanisms – Traditional subsystem

L2 diagnostic and monitoring mechanisms aim to detect runtime errors on additional platform hardware and software components following traditional functional safety practices and diagnostics techniques required by standards (see Section 3.2.2.1). However, these standards do not reflect the increasing complexity of emerging systems, therefore, L2DM mechanisms shall integrate advanced diagnostic approaches for high-performance platforms as well as the for the software applications that run on them (covered in Section 3.2.2.2).

3.2.2.1. Traditional functional safety diagnostics

Traditional functional safety standards such as IEC 61508, ISO 26262 and EN 5012x define the faults or failures that shall be considered to reach different degrees of Diagnostic Coverage (DC) (i.e., Low (60%), Medium (90%) or High (99%)) for the different system components and proposes the applicable diagnostic techniques for each: electromechanical devices, discrete hardware, bus, CPU, memories, clock, communications, sensors, final elements, etc.. The defined diagnostic strategy will comprise among others, startup diagnostics, periodic diagnostics, as well as forced checks of the safety function. On the software side, functional safety standards propose techniques and measures to prevent systematic failures on the different stages of the development life cycle (e.g., requirements specification, design and development, testing and integration…). For each technique or measure, the standard gives a recommendation (highly recommended (HR), recommended (R), not recommended (NR)) according to the target safety integrity level. Additionally, when the software implements safety functions of different safety integrity levels, it shall be demonstrated either that independence of execution is achieved both in the spatial and temporal domains, or that any violation of independence is controlled. These techniques have been widely applied over the years and are considered state-of-the-art so this report, and the SAFEXPLAIN project, do not focus on their application.

However, these standards do not reflect the increasing complexity of emerging systems, neither at hardware architectural design, nor for the software applications that run on them. Except for ISO 26262 that included a new part 11 with the topic of multicore processors in its second edition, the mechanisms described in functional safety standards focus on single-core architectures, with buses (no complex interconnects), simple memory hierarchies and no accelerators as those required to achieve the required performance in ML inference. For instance, the mechanisms defined in standards to achieve temporal independence do not generally contemplate the concurrent access to shared resources as they were not originally defined for high-performance embedded architectures with parallel access to such resources. Therefore, L2 diagnostics shall integrate traditional functional safety diagnostic mechanisms (out of the scope of this document) together with advanced diagnostic approaches for high-performance platforms explained in next subsections.

3.2.2.2. Techniques for multicore / high-performance platforms

Most HPEC platforms integrate multicore processors together with accelerators for ML inference, as it is the case of the NVIDIA Jetson Orin platform introduced in Section 1.1.2. 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. Similarly, the Certification Authorities Software Team (CAST), an international group of certification and regulatory authority representatives from the Federal Aviation Administration (FAA), provide guidance for ensuring safe implementation of multicore processing in the avionics domain [4] [9]. Next table summarizes the objectives of CAST-32A and AMC 20-193:

Table 3: Summary of CAST-32A and AMC 20-193 Objectives [9]

	ID	Description
Software Planning	PL_1	**Include MCP specific planning details in the SW plan doc.** Specific processor, number of active cores, software architecture, dynamic software features, whether it hosts an IMA-like system (with applications from different systems) or not, Robust Partitioning supported or not, methods and tools for development and verification.
Planning and Setting Resources	RU_1	**Determine configuration settings that enable to satisfy the functional, performance and timing requirements.**
	RU_2	**Critical configuration settings shall be static and protected against unintended modifications.**
	PL_2	**Include a high-level description of shared resource usage and active dynamic hardware features in the hardware and software planning documents.** Intended shared resource allocation and verification to prevent resource capabilities from being exceeded.
Inference Channels and Resource Usage	RU_3	**Identify interference channels and verify the chosen means of mitigation.** Interferences caused by shared memory, shared cache, interconnect, shared I/O or any other shared resource.
	RU_4	**Identify available resources in the intended final configuration, allocate them to the applications and verify that the demands do not exceed the available resources (under worst-case scenarios).**
Software Verification	SWV_1	**Verify that all software components function correctly and have sufficient time when all the software is executing in the intended final configuration.** Depends on the platform classification: 1. Platforms with Robust Partitioning: SW verification and WCET analysis can be done separately for each SW app. 2. All Other Platforms: If interference is mitigated for any software component or set of requirements, the verification of such components can be done separately. Otherwise, verification and WCET analysis shall be done with all software components executing together.

The objectives in Table 3 can be summarized into the following four high level principles:

1. Determining the final configuration. The designer shall determine which is the intended _final configuration that will enable to satisfy system requirements (PL_1, RU 1) and this configuration shall be protected against modification at runtime (RU 2).

2. Managing interference channels. It is required to identify interference channels in the intended final configuration and to define the means to either avoid interference by design or upper-bound it so that timing deadlines are not exceeded (RU 3). Upper-bounding interference involves analysing the use of shared resources and designing the means to control contention (PL 2).
3. Verifying the use of shared resources. Resource usage and data and control flows among cores shall be verified by guaranteeing that the software does not exceed the use of available resources even in worst-case scenarios (RU 4, SWV 1 and SWV 2).
4. Error Management. The system shall include features for multicore specific error detection and handling.

In order to achieve these four principles, techniques and measures shall be defined to ensure freedom from interference at platform level, with the support of the following L2 diagnostic and monitoring (L2DM) mechanisms:

- L2DM Configuration check: The configuration shall be defined and verified during system development process. Then, at runtime, L2DM checks that this configuration is kept (e.g., by a CRC check). Besides performance and functional correctness, also *Freedom from interference* can be highly dependent on this configuration:
  - Regarding spatial independence, most platforms usually provide memory management support allowing to set memory regions with specific permissions, as well as hardware support, such as Memory Management Units (MMUs) allowing to enforce those permissions. Therefore, appropriate operating system support will allow realizing spatial partitioning.
  - Regarding temporal independence, the platform could also provide explicit support to realize it. In particular, we aim at bounding the maximum interference that a software component might cause on the execution of another, building on the existing platform support, which provides both, means to achieve some degree of partitioning (e.g., cache partitioning), as well as means to measure interference whenever partitioning is not enough. When robust partitioning cannot be achieved by the configuration, interference channels and shared resources shall be managed as required by CAST-32A / AMC 20-193. This is covered in next point.
- L2DM Interference mitigation and control: This L2DM mechanism addresses both “managing interference channels” and “verifying use of shared resources” principles from CAST-32A / AMC 20-193. L2DM is the responsible to check that all critical platform SW components meet with their deadline and if this is the case, it refreshes an external watchdog (L3). In this way, if any critical software component violates a timing deadline or if the platform is not able to run L2 diagnostics, the L3 external device will be able to detect it. To this end, it is required that L2 knows the deadline of each critical software component, which shall be determined at design time by WCET analysis. The timing analysis strategy adopted in SAFEXPLAIN, has specific implications on L2DM. Two main scenarios have been considered:
  - The WCET of each component takes into account worst possible interference for a given platform configuration. If at design time it is verified that under worst-case interference the WCET of the critical software component is below its deadline, then at runtime the L2DM is only checking that this condition is always preserved at runtime. Next Figure 6 represents this, assuming there is a critical software component ‘A’, at design time the $WCET_A$ is determined for a given design time configuration ($CONF_D$). Then at runtime, L2DM checks that the actual platform configuration ($CONF_R$) is consistent with the

design time configuration and that execution time of SW component A (ET_A) is always below its deadline. This approach applies to all critical components in the platform. If all of them meet the deadline and all other traditional functional safety diagnostics of L2DM are passed, then L2DM refreshes an external watchdog that can be part of L3DM.

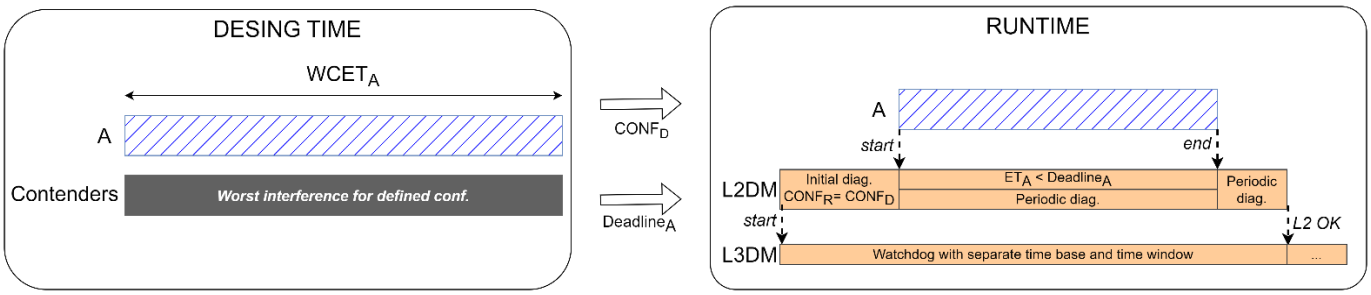


Figure 6: WCET of each component considers worst possible interference for a given platform configuration.

Following this approach, the resulting WCET estimates are usually over pessimistic, as the worst possible interference in such complex platform architectures can be very high. This can be reduced either by optimizing the configuration so that interferences are minimized or by the second approach explained below.

- The WCET of each component depends on applications running concurrently for a given platform configuration and is therefore only valid for a specific setup or scenario. This WCET estimation assumes an upper bound of the interference that the critical component can support when it is being executed. This is depicted in Figure 7. In this case, at design time, apart from the design time configuration ($CONF_D$) and the WCET of component A with limited interference ($WCET_{A\_Bounded}$), the interference limit of contenders used to estimate that WCET shall be determined too ($InterferenceLimit_A$). At runtime, L2DM follows the same approach as in the previous case and it additionally checks that contenders (e.g., SW component 'B' and 'C' on the right side of Figure 7) do not exceed the defined $InterferenceLimit_A$. To check this at runtime, different strategies can be used, depending on the approach used to obtain the WCET estimate at design time. For instance, the number of active contenders can be monitored, or the specific usage (e.g., RU_B and RU_C in Figure 7) of given shared resources (e.g., caches, memory, interconnects…).

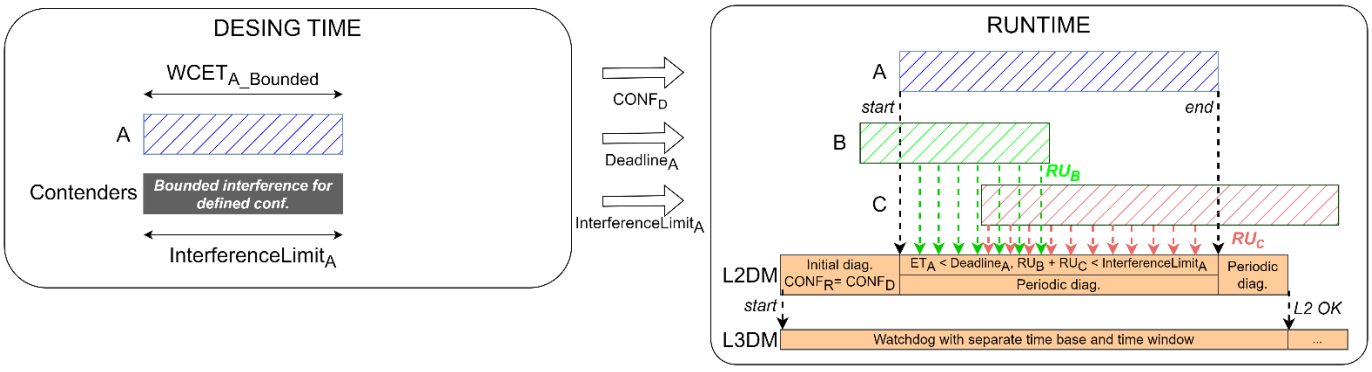


Figure 7: WCET of each component depends on applications running concurrently for a given platform configuration (valid only for a specific setup or scenario).

- L2DM Health Management: L2DM includes an error management module to react to different platform level errors. To this end, L2DM collects the results from L0 and L1

diagnostic and monitoring mechanisms. Whenever L0, L1 or L2 diagnostics and monitoring detects an error in the platform, either by startup diagnostic tests or periodic checks, L2DM is the responsible of triggering the required reaction. This will depend on the specific error and the application itself. For instance, in a TMR architecture if L0 determines that one of the redundant instances is providing incorrect inputs, the system could keep working for a limited time in a degraded mode with the other 2 active redundant instances. Similarly, if the interference mitigation and control of L2DM determines that a contender is using a shared resource more than planned in the design, the action could be to force it to stop so that it does not cause uncontrolled interferences to the critical component. In many other cases, the reaction could be to move the system to the safe state. In addition, the health monitoring in L2DM shall provide an interface with L3DM (e.g., refreshing a watchdog), in such a way that if there is a problem in the platform that affects the correct execution of L2DM, the external L3DM will be able to detect it.

3.2.3. L3 mechanisms – External

L3 diagnostics and monitoring mechanisms comprise traditional external safety mechanisms required by functional safety standards IEC 61508 / ISO 26262 / EN 5012x (e.g., external watchdog, time and power monitoring, external monitoring unit…). Moreover, the L3 diagnostics and monitoring implements error management for the lower-level diagnostic and monitoring mechanisms (i.e., L0, L1 and L2 mechanisms).

The definition and implementation of traditional functional safety mechanisms is out of the scope of the project and will therefore not be covered within this safety concept.

3.3. Supervision function

The supervision function applies elements of control theory to minimally bound AI operations. All in all, it shall control that the AI system works within a predefined safety envelope. To this end, the supervisor function shall determine a safe subset of the action space or safe envelope (prior to the execution or at runtime) and compute a set of constraints or limits. The supervision function comprises techniques such as:

- A non-AI safety function that computes an acceptable range of outputs for a given input and limits the intelligent control output (this approach might not be appropriate for systems with dynamics).
- Defining minimal bounds though control theory methods (e.g., barrier functions approaches).
- Checking functions such as metrological self-check or self-validation.

In addition, the supervision function implements explainability techniques to provide understanding on the DL model operation as well as on the decisions, recommendations, or predictions it provides.

3.4. Traditional subsystem

By traditional subsystem we refer to a subsystem that does not include any ML model according to the system decomposition of Figure 1. In the scope of the reference architecture, this traditional subsystem can be either:

- Functions complementary to the ML constituent, required to perform the function required by it.
- Fallback subsystem: Following known proper functional safety precautions, a safe back-up function to the ML component might be implemented within the non-AI subsystem. This back-up decision system will take over the system in the event that a functional safety problem is detected, ensuring no harm is done by the AI system. The implementation of such a fallback function is use-case dependent and is therefore out of the scope of the project.

Part III – Railway case study

4. System concept specification

This section presents the railway domain object detector controller. It describes the Automatic Train Operation, and its architecture, relevant subsystems for the case-study and safety integrity level to achieve.

4.1. System description

The railway use-case comprises an Automatic Train Operation (ATO) system that detects obstacles in the railway and estimates their distance by the assistance of cameras (i.e., right / left cameras). According to the detected obstacle (i.e., critical / non-critical), the distance from it and its location (in the track / outside the track), the system responds accordingly (i.e., warn the driver, activate service brake, activate emergency brake). All in all, the system implements a safety function(s) to minimize the risk of the train running over or damaging persons in the track, colliding with obstacles in the track, or damaging passengers on the train itself.

Following the incremental strategy for the adoption of AI that has been defined in Section 2.2, the following situations have been considered, where the AI system plays different roles within the ATO system:

a) The simplest case where the AI system is not part of the safety function (control system). The AI system detects an obstacle (another train, person, car…) in the tracks and warns the driver of the potential danger. The driver must act safely activating the brakes if necessary. He is the only safety responsible (see Figure 8).

b) The second case is where the AI system participates in the safety chain. But this participation has a low safety integrity level. The AI system detects an obstacle in line of collision, and it warns the train driver of the potential hazard. Meanwhile, the AI system activates the service brake reducing the speed of the train. Service brake corresponds to SIL2 safety function. The alerted driver can activate the emergency brake, SIL 4 safety integrity level, or temporarily deactivate the service brake (SIL 2) (see Figure 9).

c) The third case is where the AI system is part of the safety chain. The AI system detects an obstacle in line of collision and warns the control centre. The AI system activates the service brake (SIL 2) which starts to reduce the speed. Besides, it calculates the braking distance with the current braking capacity, activating the emergency brake (SIL 4) when the collision distance is below a threshold. That is, avoiding the collision, but only activating it if the hazard is not resolved yet (see Figure 10).

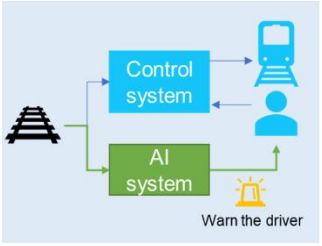


Figure 8: AI system for warnings

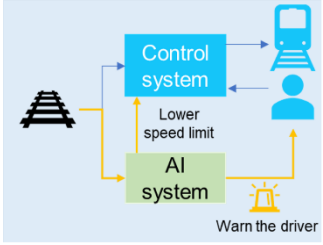


Figure 9: AI system participating with a low SIL (e.g. SIL 1 / SIL 2)

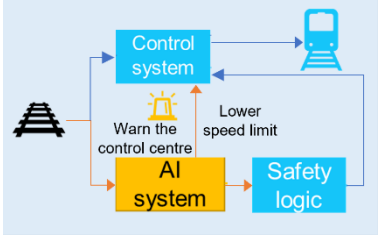


Figure 10: AI system being part of the safety chain and actuating with a high SIL (e.g., SIL 3 / SIL 4).

In the table below (see Table 4) the train operation basic functions of the three cases have been classified considering their grade of automation, usage level and AI technology class.

Table 4: GoA levels of the system

Grades and type of automation		Basic functions of train operation		Use case	AI technology class & AI Application and Usage Level defined by ISO/IEC TR 5469	Applicable SP
Grades of Automation	**Type of train operation**	**Supervise track (prevent collision)**	**Driving (control acceleration and braking)**	**Description**		
GoA 1	Non automated train operation (NTO)	Driver	Driver	The IA algorithm is not part of the safety function. Detects an obstacle in the tracks and warns the driver. The driver (safety responsible) must act safely (activate brakes).	Class II Usage Level D	SP1
GoA 2	Semi-automated train operation	Driver	Automatic	The IA algorithm participates in the safety chain (at a low-level safety integrity). Detects an obstacle in line of collision and warns the driver. Activates the service brake (SIL 2). The alerted driver (safety responsible) can activate the emergency brake (SIL 4) or deactivate the service brake (SIL 2).	Class II Usage Level C	SP2
GoA 3	Driverless train operation	Automatic	Automatic	The IA algorithm is part of the safety chain. Detects an obstacle in line of collision and warns the control centre (only for GoA 3). Activates the service brake (SIL 2) and calculates the braking distance with the current braking capacity, activating the emergency brake (SIL 4) when the collision distance is below a threshold.	Class III (Class II if explainability techniques are applied) Usage Level A1	SP3
GoA 4	Unattended train operation (UTO)	Automatic	Automatic			

Figure 11 shows the high-level architecture of the ATO system, which consists of the following main components:

- Object Detection: Detects objects[2] in the two images given by the left and right cameras and identifies the train tracks.
- Distance Estimation: Using stereo vision, distance is estimated for all the objects detected in the two images given by the left and right cameras.
- Object classification and positioning: classifies the objects detected by the 'object and track detection' module according to their criticality and their position, discarding non-critical objects and objects outside the track. Then, the depth estimated by the 'distance estimation' module is compared against the defined threshold(s) and according to the grade and type of automation of the train, different actions are taken:
  - GoA 1: warn the driver if an obstacle is detected on the tracks.
  - GoA 2: warn the driver if an obstacle is detected on the tracks and activate the service brake (SIL 2).
  - GoA 3 and GoA 4: warn the driver if an obstacle is detected on the tracks and activate the service brake (SIL 2). Activating the emergency brake (SIL 4) when the collision distance is below a predefined threshold.

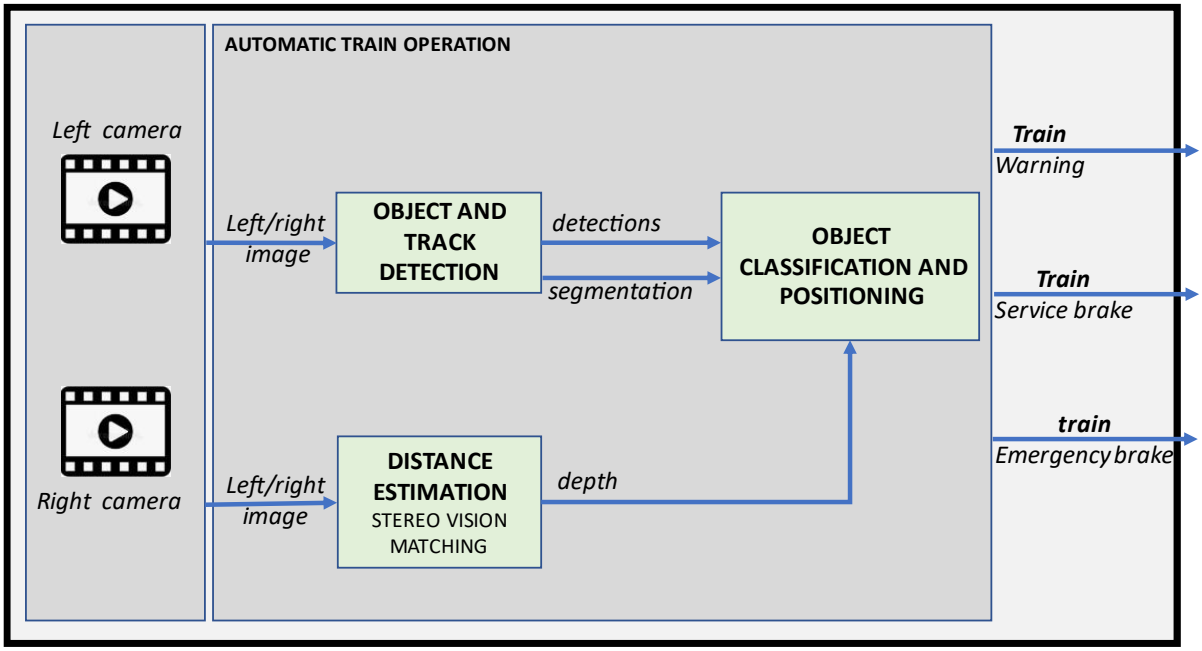


Figure 11: High-level SW architecture of the system

Beside the AI-based sub-system, a traditional train control sub-system shall implement the algorithm that based on the driver commands controls the train operation (i.e., set train speed, activate/deactivate service brake, activate/deactivate emergency brake). In the following subsections, this is referred as the train operation control sub-systems. However, given that this is a traditional software component, its safety implications are considered to be out of the scope of the project.

[2] The objects to be detected shall be defined in the ODD.

5. Safety Patterns

Following the incremental strategy described in Section2.2, this section presents the mapping of the railway use-case (see Figure 11) to the different DL usages levels, based on the grade and type of automation of the system (see Table 4). In particular, three Safety Patterns are defined:

- Safety Pattern 1 (SP1) (for DL usage level D / EASA Level 1).
- Safety Pattern 2 (SP2) (for DL usage level C / EASA Level 2).
- Safety Pattern 3 (SP3) (for DL usage level A1 / EASA Level 3).

The following subsections present, for each Safety Patterns, the proposed system architecture and safety techniques, as well as its application on the project target platform (i.e., NVIDIA Orin). The description of the Safety Patterns follows an incremental approach; therefore, each Safety Pattern comprises also the safety techniques of the previous one.

5.1. Safety Pattern 1 (SP1)

On this safety pattern, the AI subsystem aids the driver or person responsible of the safety of the system. However, the AI subsystem is not part of the safety function.

On the following subsections we present the system architecture for SP1 as well as the corresponding diagnostic and monitoring mechanisms.

5.1.1. System Architecture

Figure 12 shows the mapping of the GoA1 non automated train operation (NTO) railway use-case presented in Section 4 to the Safety Pattern 1. This pattern assumes the integration of safety related software (i.e., train operation control, safety diagnostics) and non-safety related software (i.e., object and track detection, distance estimation, object classification and positioning, supervision function) within the same SoC on a High-Performance Computing platform. The main components of the reference safety architecture are applied as follows in SP1:

- Diverse Redundancy: In this safety pattern, the safety function is implemented by the train operation control non-AI subsystem, therefore, the AI/ML constituent, running the object and tract detection, has no safety implications. Given that, diverse redundancy is not required in the AI/ML constituent.
- Diagnostic and monitoring mechanisms: Even if the AI/ML constituent is not safety related, in SP1 the main challenge rests on the integration of both the safety software together with the software involved in the inference of the DL model and to guarantee freedom from interference among them. Therefore, L1DM is the responsible for monitoring resource usage and timing of the AI/ML constituent, and L2DM / L3DM to apply the required diagnostic mechanisms on the train operation control safety subsystem and guarantee freedom from interference. The application of the diagnostic and monitoring strategy in SP1 is further elaborated in Subsection 5.1.2.
- Supervision function: the supervisor has no safety implications, and its purpose is to improve the explainability of the AI-based subsystem.
- Traditional subsystem: in SP1 the safety function is performed by the train operation control traditional SW component. Moreover, the traditional subsystem comprises non-safety related components that complement the object and track detection AI component (i.e.,

distance estimation and object classification and positioning). No fallback subsystem is considered.

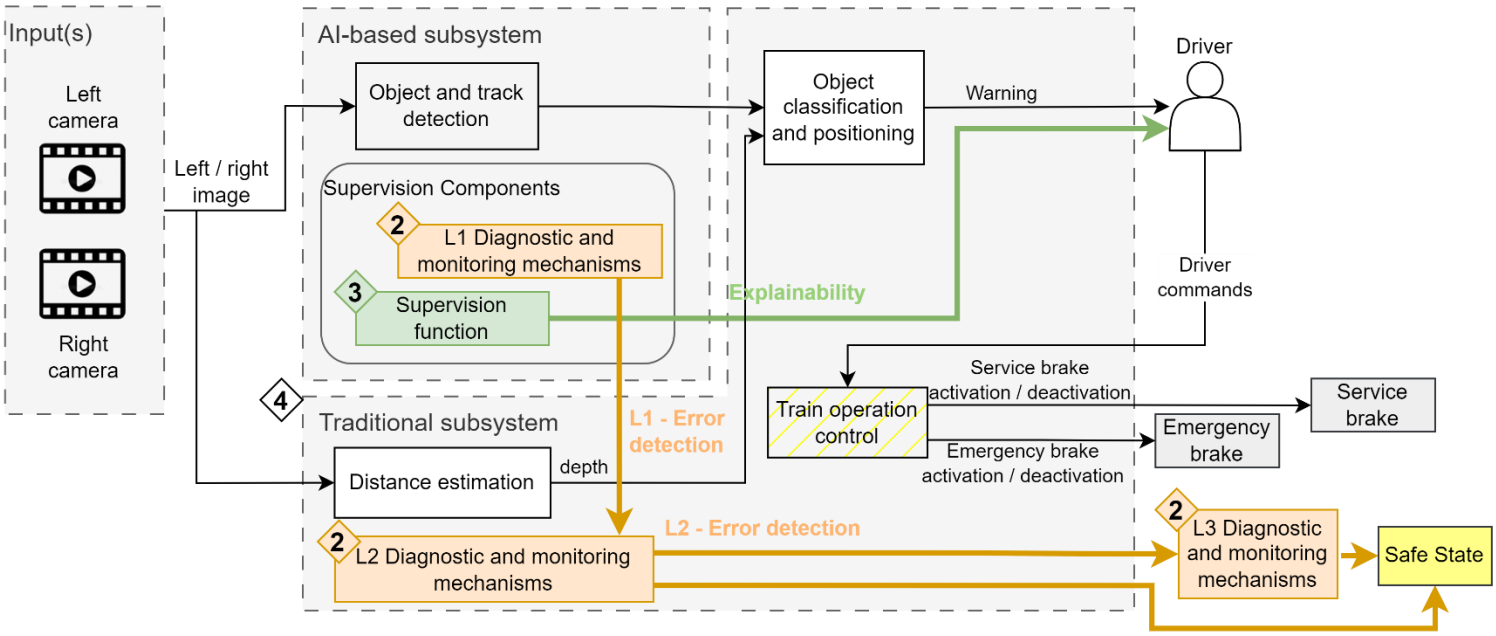


Figure 12: Non automated train operation (NTO) – Conceptual System Architecture (Safety Pattern 1)

5.1.2. Diagnostic and monitoring mechanisms

Next subsections explain the application of the hierarchical diagnostics and monitoring approach of Section 3.2 to SP1 on the SAFEXPLAIN platform.

5.1.2.1. L1DM – SAFEXPLAIN PLATFORM

As the AI subsystem is not safety related, there are no specific diagnostics for it. However, the L1 monitoring mechanisms are necessary to guarantee the required freedom from interference at platform level. In this particular case, the monitoring of the use of shared resources among the safety and non-safety software components (i.e., memory, caches, interconnect) is of special interest. In SP1, L1DM will collect such events for the non-safety AI subsystem and provide the information to the L2DM.

5.1.2.2. L2DM – SAFEXPLAIN PLATFORM

The L2DM mechanism implements traditional functional safety techniques and measures according to the recommendations on functional safety standards. These diagnostics comprise start-up as well as periodic checks that can be applied to the different subsystems that participate in the execution of the safety function (e.g., sensors, CPU, memories, clock, electromechanical devices, etc.). To this end, NVIDIA provides support documentation as well as libraries for the implementation of safety diagnostics.

In addition, L2DM is responsible for checking system configuration and providing the mechanisms for interference mitigation and control. As introduced in Section 1.1.2, the NVIDIA Orin platform provides a number of diverse processing resources (CCPLEX core clusters, SPE, GPUs…) with multiple configuration options. The allocation of such resources to software components and their configuration plays a crucial role in the mitigation of interference required by SP1. Figure 13 and Table 5 present different resources and configuration options that could be adopted for SP1.

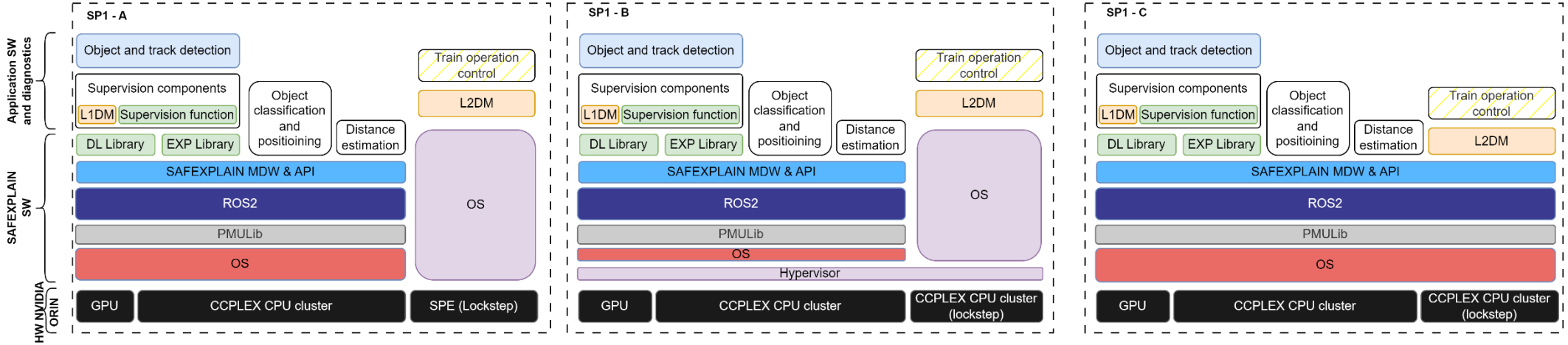


Figure 13: SP1 to NVIDIA Orin resource allocation and configuration options

Table 5: SP1 to NVIDIA Orin resource allocation and configuration options

SP1 Element	Safety / non-safety	SP1 - A NVIDIA Orin resources and configuration	SP1 - B NVIDIA Orin resources and configuration	SP1 - C NVIDIA Orin resources and configuration
Object and tract detection	Non-safety AI based SW	CCPLEX CPU Cluster (Cortex A78) GPU for AI inference Memory controller fabric and traffic from CPU cluster to GPU MMUs for spatial independence SAFEXPLAIN SW Stack	CCPLEX CPU Cluster (Cortex A78) – different CPU cluster for safety SW GPU for AI inference Memory controller fabric and traffic from CPU cluster to GPU MMUs for spatial independence L4 cache partitioning or disabled SAFEXPLAIN SW Stack separated from safety SW by hypervisor	CCPLEX CPU Cluster (Cortex A78) – different CPU cluster for safety SW GPU for AI inference Memory controller fabric and traffic from CPU cluster to GPU MMUs for spatial independence L4 cache partitioning or disabled SAFEXPLAIN SW Stack
Supervision components	Non-safety traditional or AI based SW			
Distance estimation	Non-safety traditional SW			
Object classification and positioning	Non-safety traditional SW			

Train operation control (safety function)	Safety traditional SW	SPE in lockstep configuration No sharing of caches with CCPLEX MMUs for spatial independence Safe RTOS on top of SPEs	CCPLEX CPU Cluster (Cortex A78) in lockstep – different CPU cluster for safety SW MMUs for spatial independence L4 cache partitioning or disabled Safe RTOS on top of hypervisor	CCPLEX CPU Cluster (Cortex A78) in lockstep – different CPU cluster for safety SW MMUs for spatial independence L4 cache partitioning or disabled SAFEXPLAIN SW Stack
L2DM	Safety traditional SW			
Pros		Highest independence: different processing elements, less shared resources, different OS and SW stack.	Better performance while preserving independence: different CPU clusters with partitioning approaches, different OS and SW stack.	Ease of integration, same SW stack for all platform elements.
Cons		Limited performance on SPEs.	Need of a hypervisor.	Less independence, required safety guarantees on the SAFEXPLAIN SW Stack.

The three SP1 configuration options rely on mechanisms provided by the platform, which include cache partitioning features in the CCPLEX, as well as performance monitoring counters allowing to monitor resource usage in the CCPLEX. So far, the monitoring capabilities of the shared L4 cache and the GPU need to be analysed, and freedom from interference in those resources may need to resort to software support such as, for instance, task scheduling constraints to avoid interference by construction as much as possible.

- **Independent computing components**: The NVIDIA Orin provides abundant computing resources, the safety critical tasks can be allocated either to the SPE or to a specific CCPLEX cluster, increasing independence and diversity among safety and non-safety tasks.
- **Spatial independence:** As explained before, the NVIDIA Orin platform provides explicit support to enable spatial independence in the form of MMUs. Therefore, spatial independence can be achieved by design with usual operating system support.
- **Temporal independence:** Safety related tasks needing temporal independence can be executed in the SPE or CCPLEX. The degree of independence achieved across tasks running in the same CCPLEX core cluster is relatively high if L3 cache partitioning is used, and even higher across tasks running in different core clusters as proposed in SP1 - B and SP1 - C. The remaining interference can be measured with existing event monitors so that safety measures can be built at software level if measured interference reaches system or application-specific thresholds.

- **Support for lockstep redundancy:** As explained before, core clusters in the CCPLEX and the SPE provide lockstep support. Hence, safety related tasks requiring such redundancy can be deployed on the SPE or the CCPLEX by properly configuring the core cluster where they run.

Following the approach described in Subsection 3.2.2.2, determining the configuration at system design time is crucial to perform the timing analysis and interference mitigation and control. Based on the configuration, the timing and interference bounds will be computed and at runtime, L2DM will check that the configuration corresponds to that defined at design time (e.g., through a CRC check).

5.2. Safety Pattern 2 (SP2)

In the GoA2 semi-automated train operation case-study, the AI-based subsystem becomes a safety-critical component, since the IA object and track detection algorithm participates in the safety chain (at a low-level safety integrity). Therefore, in addition to the mechanisms presented in SP1 to guarantee the independence among safety and non-safety software components, new mechanisms are required to ensure the correct behaviour of the object and track detection AI-based subsystem. In the semi-automated train operation, if the AI-based subsystem detects an obstacle in line of collision it warns the driver and activates the service brake (SIL 2). Therefore, the AI-based safety function has a SIL 2 according to IEC 61508. Whereas the train operation control will activate the emergency brake (SIL 4) or deactivate the service brake (SIL 2) following the driver commands.

5.2.1. System Architecture

Figure 14 shows the mapping of the GoA2 semi-automated train operation use-case presented in Section 4 to the Safety Pattern 2. This pattern assumes the integration of mixed-criticality software, SIL 2 software (i.e., object and track detection, decision function, supervision components, distance estimation, object classification and positioning) and SIL 4 software[3] (i.e., train operation control, L2DM mechanisms) within the same SoC on a High-Performance Computing platform.

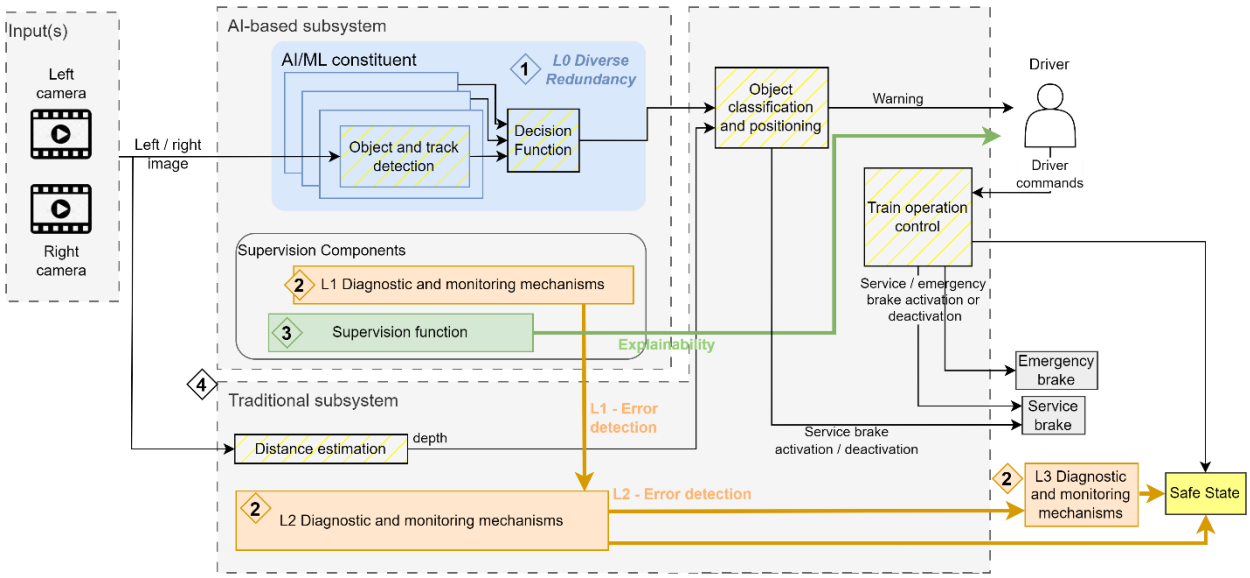


Figure 14: Semi-automated train operation – Conceptual System Architecture (Safety Pattern 2)

The specific safety mechanisms that apply and the way of addressing them can vary according to the implications of the integrity level of the safety functions. As previously mentioned, on the GoA2 semi-automated train operation use-case the object and track detection AI-based function

[3] The safety implications of the traditional SW are considered to be out of the scope of the project. Therefore, this safety concept does not analyse how SIL 4 is achieved for the train operation control. However, off-chip redundancy would be required to meet with the safety requirements associated to a SIL 4.

interacts participates in the control of the service brake, which corresponds to SIL 2 safety function. Next Table 6, shows the architectural implications of this integrity level according to IEC 61508-2:

Table 6: Maximum allowable safety integrity level according to IEC-61508-2 for Type B safety related elements (SP2)

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

Therefore, semi-automated train operation requires a high diagnostic coverage (90-99% for SIL 2) or increasing the Hardware Fault Tolerance (HFT). However, achieving this HFT > 0 on the same chip has several implications on the hardware silicon level (IEC 61508-2 Annex E) which shall be guaranteed by the platform manufacturer. As the SAFEXPLAIN platform does not provide such guarantees, off-chip redundancy would be required, which is the case of SP3. Whereas the focus or SP2 is on HFT = 0 with low to medium diagnostic coverage.

Taking this into account, the main components of the railway use-case are applied as follows in SP2:

- **Diverse Redundancy:** On chip redundancy approaches are considered to improve the safe failure fraction and diagnostic coverage of the object and track detection function running in the AI/ML constituent, further explained in Subsection 5.2.2.
- **Diagnostic and monitoring mechanisms:** Same mechanism as in SP1 apply, with additional diagnostics for the object and track detection function running in the AI/ML constituent in the L1DM further elaborated in Subsection 5.2.3.
- **Supervision function:** A supervision function according to the description in the reference architecture shall be integrated too. The supervision function does not have any particularities for SP2, so the main description in Section 3.3 can be checked.
- **Traditional subsystem:** the traditional subsystem comprises the distance estimation and object classification and positioning SIL 2 safety functions. Moreover, the train operation control SIL 4 is also part of the traditional subsystem.

5.2.2. Diverse redundancy

From Table 2, diverse redundancy schemes with low or medium DC can be applied in this safety pattern, based on the specific requirements of the application (e.g., DSR_1, DSR_2, DSR_3, DSR_4). These techniques have been already explained in Section 3.1.

For the implementation of diverse redundancy in the SAFEXPLAIN platform, AI tasks often need the use of GPUs and/or AI accelerators. Depending on the diverse redundancy scheme, the redundant instances can either run redundantly on the GPU or using diverse processing resources such as GPU and CPU, or GPU and other AI accelerators (DSR_2). The NVIDIA Orin platform allows multiple kernels to run simultaneously. However, the GPU does not provide explicit support for diverse redundancy, which needs to be built by external means using the techniques described before. Those techniques provide explicit support to manage random hardware faults affecting one of the redundant instances. However, faults potentially leading to common cause failures

need additional support. If computation can be identical across redundant instances (e.g., some results are obtained applying the same operations on identical data), we may need mechanisms to enforce the use of separated computing resources and to make those computations not to occur simultaneously, as in the case of lockstep cores. We aim at realizing such features, if eventually needed, resorting to techniques we already devised in the past and realized in both, NVIDIA and Intel GPUs [10] [11]. In any case, we note that, as previous work noted, the GPU in the NVIDIA Orin platform includes some unique components, such as the hardware scheduler, which needs to be used by redundant tasks. Hence, with the information available of the platform, we have to assume that some permanent faults could potentially lead to identical errors for redundant kernels, and hence, they could not be directly detected with the solutions described here. In general, such type of errors should be managed at system level by, for instance, deploying diverse object detection mechanisms (e.g., based on multiple independent cameras, or on other sensors such as radars and LIDARs).

5.2.3. Diagnostic and monitoring mechanisms

Next subsections explain the application of the hierarchical diagnostics and monitoring approach of Section 3.2 to the GoA2 semi-automated train operation system to the SP2 on the SAFEXPLAIN platform.

5.2.3.1. L1DM – SAFEXPLAIN PLATFORM

L1DM diagnostics mechanisms from Section 3.2.1 can be selected and evaluated to reach the required diagnostic coverage for SP2.

To reduce common cause failures among redundant instances in the AI/ML constituent, their independence shall be guaranteed. Therefore, L1DM will monitor the use of resources and other relevant events for each instance of the redundant architecture and share the information with L2DM for a platform level monitoring of all system components.

5.2.3.2. L2DM – SAFEXPLAIN PLATFORM

Following the incremental approach, L2DM platform level diagnostics shall be applied in the same way as for SP1. In this case, the NVIDIA Orin platform resources used for the AI/ML constituent become safety relevant too, which would require additional diagnostics with respect to SP1 and the most suitable platform configurations shall be adapted too. In addition, since the system includes application with different integrity levels (i.e., SIL 2 and SIL 3 software), independence of execution shall be guaranteed following the same approach as in SP1. Figure 15 presents a configuration option that could be adopted for SP2, which could be combined with other solutions shown in SP1 (see Figure) as described in Table 7.

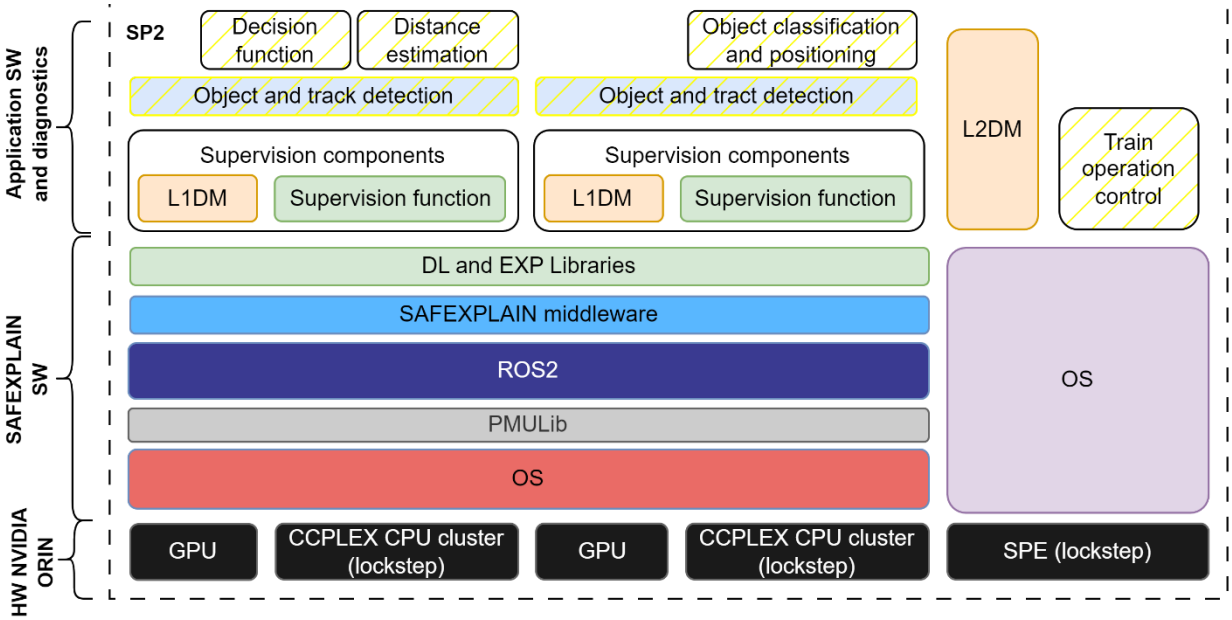


Figure 15: SP2 to NVIDIA Orin resource allocation and configuration options

Table 7: SP2 to NVIDIA Orin resource allocation and configuration options

SP2 Element	Safety / non-safety	SP2 - A NVIDIA Orin resources and configuration
Object and track detection	AI based SIL 2 SW	Two instances, each in one separate CCPLEX CPU Cluster (Cortex A78) in lockstep configuration GPU for AI inference (depending on the DRS CPU or other computing resources could also be used to improve diversity) Memory controller fabric and traffic from CPU cluster to GPU MMUs for spatial independence SAFEXPLAIN SW Stack
Supervision components	Traditional or AI based SIL 2 SW	Each AI/ML constituent has each own L1DM and optionally each own supervisor function (depends on user application). Depending on the implementation of the supervision component, it may need GPUs for improved performance (e.g., AI based supervision function). The supervision components can share same CCPLEX CPU Cluster (Cortex A78) in lockstep configuration as the AI/ML constituent. MMUs for spatial independence SAFEXPLAIN SW Stack
Decision function	Traditional SIL 2 SW	These SW components can run on any of the CCPLEX CPU Cluster (Cortex A78) in lockstep configuration used for the AI/ML constituent with the same configuration, since they have the same integrity level.
Distance estimation		
Object classification and positioning		
Train operation control	Traditional SIL 4 SW	CCPLEX CPU Cluster (Cortex A78) or SPE in lockstep configuration. MMUs for spatial independence L4 cache partitioning or disabled SAFEXPLAIN SW Stack or different OS on top of SPEs or hypervisor
L2DM	Traditional SIL 4 SW	

The following mechanisms are provided by the platform:

- **Independent computing components**: In this case, all SIL 2 software can be run in CCPLEX CPUs in lockstep mode, using different clusters for the redundant replicas. Other accelerators or different computing resources could also be used to improve diversity. Moreover, all SIL 3 software can run in an independent computing component such as the SPE or an independent CCPEX CPU cluster in in lockstep configuration.
- **Spatial independence:** The same support for SP1 addresses the needs of SP2.
- **Temporal independence:** The same support for non-AI components on SP1 addresses the needs of SP2 to provide temporal independence between components with different integrity levels. Regarding AI components, they are expected to use the GPU and/or the AI accelerators. Those accelerators have been devised to maximize throughput. However, either they do not support concurrency (e.g., AI accelerators) or support it without specific temporal independence support (e.g., GPU). Hence, AI tasks needing concurrency will have a high degree of temporal isolation if they use different accelerators. If they are deployed on the same one (i.e., the GPU) and run simultaneously, timing interference can be arbitrarily high making its monitoring and control unfeasible. Therefore, those tasks need to be run sequentially if they need the very same accelerator.
- **Support for lockstep redundancy:** The same support for non-AI components on SP1 addresses the needs of SP2.

5.3. Safety Pattern 3 (SP3)

Safety pattern 3 follows the same approach as SP2 but it is assumed that the system has a higher degree of autonomy and therefore the safety implications of the AI-based subsystem are higher. In fact, the GoA 3 driverless train operation and the GoA 4 Unattended Train Operation (UTO) shall be mapped to this SP. Therefore, we consider a safety integrity level of SIL 4 according to IEC 61508 for both, GoA 3 driverless train operation and the GoA 4 UTO applications.

5.3.1. System Architecture

In order to achieve higher integrity levels (with respect to SP2), SP3 involves either providing a high diagnostic coverage (≥99%) or increasing the HFT as shown in Table 8 according to IEC 61508-2.

Table 8: Maximum allowable safety integrity level according to IEC-61508-2 for Type B safety related elements (SP3)

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

However, achieving a ≥99% DC in complex platforms such as the NVIDIA Orin considered in SAFEXPLAIN is very challenging and would highly depend on the guarantees provided by the platform manufacturer. The complexity of the SW stack and lack of transparency on the platform further exacerbate this. Therefore, we assume that for SP3 an HFT > 0 is required. To this end, off-chip redundancy shall be applied due to the restrictions of IEC 61508-2 Annex E as already

mentioned in SP2. Figure 16 shows an example of SP3 implementation based on a TMR architecture, depending on the implementation HFT = 1 (2oo3) or HFT = 2 (1oo3) can be achieved.

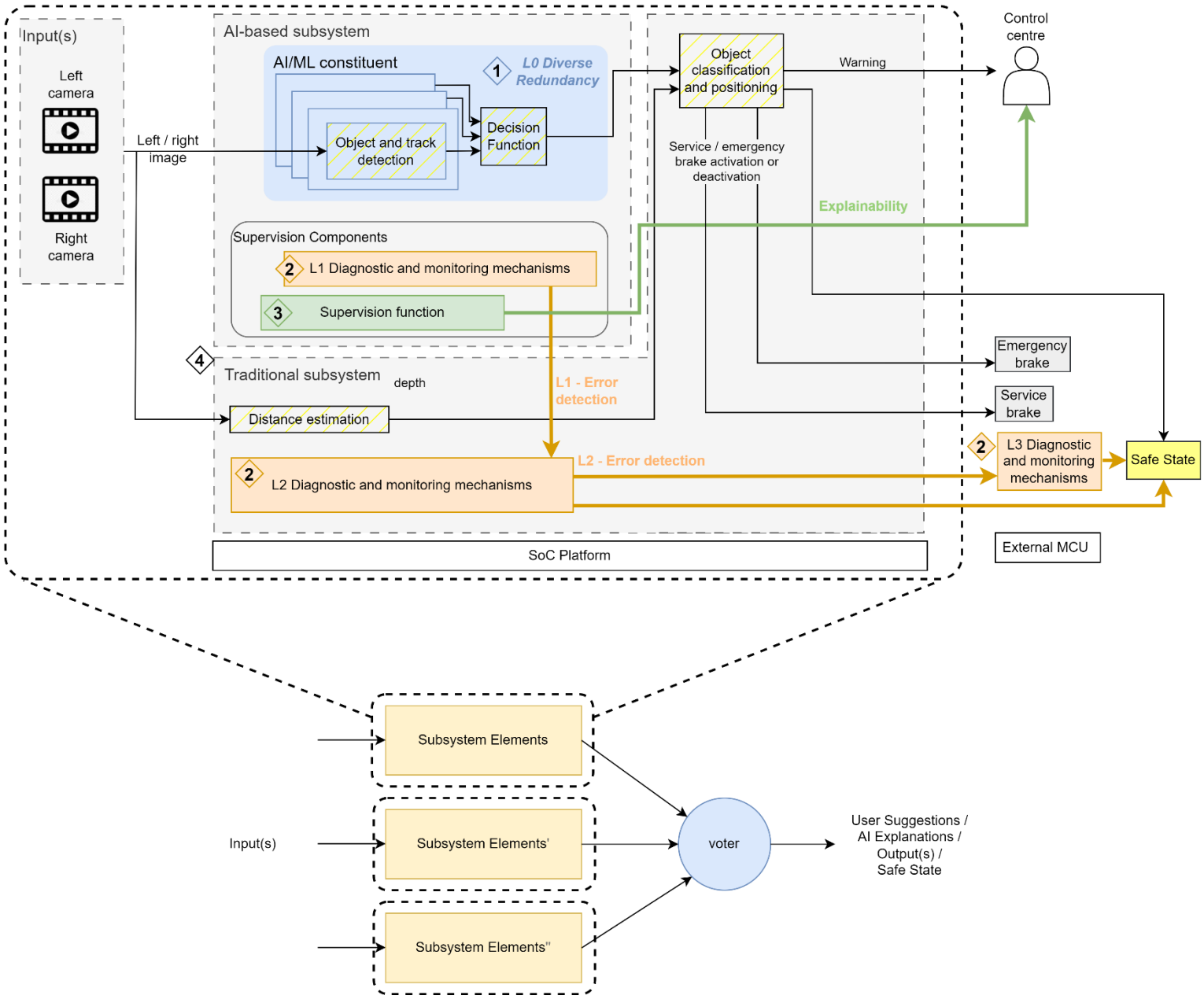


Figure 16: Driverless train operation and unattended train operation (UTO) – Conceptual System Architecture (Safety Pattern 3)

If we look at each individual replica of the TRM implementation, each node could be implemented following SP2 safety pattern:

- **Diverse Redundancy:** In SP3 off-chip redundancy is applied. Therefore, the voter should combine the outputs from different computing nodes. Each computing node can have its own voter in such a way that if any node detects discrepancies, the safe state could be reached (1oo3 – HFT =2). Alternatively, the voting can be done in an external element (e.g., external MCU) and apply majority voting, meaning that at least 2 of the 3 redundant nodes shall provide the same output (HFT = 1). In order to apply this off-chip redundancy, diverse redundancy schemes with medium to high DC can be applied in this safety pattern, based on

the specific requirements of the application (e.g., DSR_4, DSR_5, DSR_6). Off-chip redundancy can be combined with on chip redundancy approaches to improve the safe failure fraction and diagnostic coverage of each redundant node, following the same solutions as in SP2.

- **Diagnostic and monitoring mechanisms:** In each redundant instance the same solutions as for SP2 apply in SP3. L3DM would now monitor the status of all the redundant replicas.
- **Supervision function:** Each redundant replica of SP3 can implement a supervisor function according to the description in the reference architecture in Section 3.3.
- **Traditional subsystem**: the traditional subsystem comprises the distance estimation and object classification and positioning SIL 4 safety functions.

Acronyms and Abbreviations

AI	Artificial Intelligence
ATO	Automatic Train Operation
CAST	Certification Authorities Software Team
CBOD	Camera Based Object Detection
CPU	Central Processing Unit
DC	Diagnostic Coverage
DL	Deep Learning
DMR	Dual Modular Redundancy
DNN	Deep Neural Network
DRS	Diverse Redundancy Schemes
EASA	European Aviation Safety Agency
FAA	Federal Aviation Administration
FSM	Functional Safety Management
FUSA	Functional Safety
GPU	Graphics Processing Unit
HFT	Hardware Fault Tolerance
HPEC	High-Performance Embedded Computing
IoU	Intersection over Union
L1DM	L1 Diagnostics and Monitoring
L2DM	L2 Diagnostics and Monitoring
L3DM	L3 Diagnostics and Monitoring
ML	Machine Learning
MPSoC	Multi-Processor System-on-Chip
NMS	non maximum suppression
ODD	Operational Design Domain
OS	Operating System
SIL	Safety Integrity Level
SM	Stream Multiprocessor
SoC	System on Chip
SP	Safety pattern
SPE	Sensor Processing Engine
TMR	Triple Modular Redundancy
UTO	Unattended Train Operation
WCET	Worst Case Execution Time

References

[1] European Union Aviation Safety Agency (EASA), “EASA Concept Paper: guidance for Level 1 & 2 machine learning applications,,” 2023.

[2] International Electrotechnical Commission (IEC), “IEC 61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems,” 2010.

[3] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), “ISO/IEC TR 5469 Artificial Intelligence - Functional safety and AI systems,” 2024.

[4] European Union Aviation Safety Agency (EASA), “AMC 20-193 Use of multi-core processors,” 2022.

[5] EGAS Workgroup, “Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units. Version 5.5,” 2013.

[6] J. Fernández, I. Agirre, L. Belategi, J. Pérez-Cerrolaza, A. Adell, J. Imaz, C. Donzella and J. Abella, “AI-FSM,” 2024. [Online]. Available: https://zenodo.org/records/10964402.

[7] N. J. G. B. G. R. V. S. A. &. T. P. Humbatova, “Taxonomy of real faults in deep learning systems,” in *Proceedings of the ACM/IEEE 42nd international conference on software engineering (pp. 1110-1121)*, 2020.

[8] R. H. A. G. S. &. Z. S. Schnitzer, “AI Hazard Management: A framework for the systematic management of root causes for AI risks,” in *International Conference on Frontiers of Artificial Intelligence, Ethics, and Multidisciplinary Applications (pp. 359-375)*, Singapore, 2023.

[9] I. Agirre, J. Abella, M. Azkarate-Askasua and F. J. Cazorla, “On the Tailoring of CAST-32A Certification Guidance to Real COTS Multicore Architectures,” in *12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2017.

[10] S. Alcaide, L. Kosmidis, C. Hernandez and J. Abella, “Software-only Diverse Redundancy on GPUs for Autonomous Driving Platforms,” in *25th IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2019.

[11] N. Andriotis, A. Serrano-Cases, S. Alcaide, J. Abella, F. J. Cazorla, Y. Peng, A. Baldovin, M. Paulitsch and V. Tsymbal, “A Software-Only Approach to Enable Diverse Redundancy on Intel GPUs for Safety-Related Kernels,” in *38th ACM/SIGAPP Symposium On Applied Computing (SAC)*, 2023.

[12] International Standardization Organization, “ISO 26262. Road Vehicles - Functional Safety. Second edition,” 2018.

[13] SAFEXPLAIN Consortium, “D1.1 Requirements, Success Criteria and Platforms,” 2023.

[14] SAFEXPLAIN Consortium, “D2.1 Safety Lifecycle Considerations,” 2024.

[15] SAFEXPLAIN Consortium, “D4.1 Interim Platform Technologies Report,” 2024.

[16] SAFEXPLAIN Consortium, "D3.1 Specifiability, explainability, traceability, and robustness proof-of-concept and argumentation," 2024.

[17] R. Padilla, S. Netto and E. da-Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," in *International Conference on Systems, Signals and Image Processing (IWSSIP)*, Niteroi, Brazil, 2020.

[18] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," in *ArXiv*, 2020.

[19] M. Caro, H. Tabani and J. Abella, "At-scale assessment of weight clustering for energy-efficient object detection accelerators," in *In Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22)*, 2022.

[20] J. Heras, "Clodsa image augmentation library for object detection," Accessed Oct-2023.

[21] M. Caro, J. Fornt and J. Abella, "Efficient Diverse Redundant DNNs for Autonomous Driving," in *IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, Torino, Italy, 2023.

[22] C. Li, "YOLOv4 TensorFlow Keras implementation," Accessed Oct-2023.

[23] Lin, TY. et al., "Microsoft COCO: Common Objects in Context," in *European Conference on Computer Vision (ECCV)*, 2014.

[24] Everingham, M., Eslami, S.M.A., Van Gool, L. et al., "The PASCAL Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision,* no. 111, 2015.

[25] J. Hauser, "Berkeley SoftFloat," 2018.

[26] J. Fernandez, I. Agirre, J. Perez, F. J. Cazorla and J. Abella, "A Methodology for Selective Protection of Matrix Multiplications: a Diagnostic Coverage and Performance Trade-off for CNNs Executed on GPUs," in *6th International Conference on System Reliability and Safety (ICSRS)*, 2022.

[27] J. Fernandez, J. Perez, I. Agirre, I. Allende, J. Abella and F. J. Cazorla, "Towards Functional Safety Compliance of Matrix-Matrix Multiplication for Machine Learning-based Autonomous Systems," *Elsevier Journal of Systems Architecture (JSA),* vol. 121, 2021.

Annex B: Railway Safety Concept Review Meeting with TÜV Rehinland

SAFEXPLAIN

Safe and Explainable
Critical Embedded Systems based on AI

Safe and explainable critical embedded systems based on AI

TÜV Review Meeting (activity 2)

Irune Yarza

ikerlan
MEMBER OF BASQUE RESEARCH & TECHNOLOGY ALLIANCE

Funded by the European Union

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

Attendees

ikerlan
MEMBER OF BASQUE RESEARCH & TECHNOLOGY ALLIANCE

- Irune Yarza
- Irune Agirre
- Javi Fernández
- Jon Pérez

TÜVRheinland®
Precisely Right.

- Ralf Röhrig

exida®

- Carlo Donzella
- Giuseppe Nicosia
- Stefano Lodico

Agenda

- TÜV Rheinland collaboration
- Context
- Reference Safety Architecture
- Sources of Risk and Incremental Strategy
- Automatic Train Operation (ATO) system

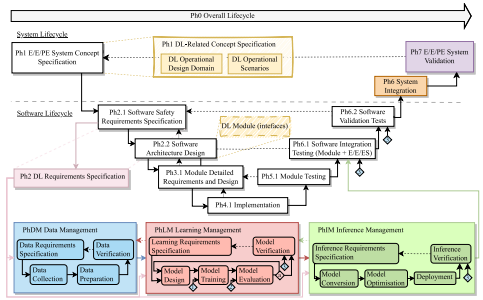
TÜV Rheinland collaboration

IKERLAN requests TÜV Rheinland to carry out the following tasks:

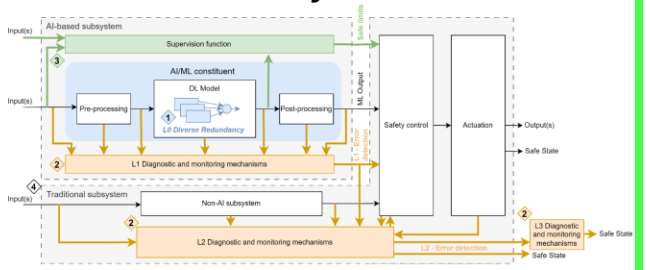
WP 0	Virtual kick off meeting to introduce the project and the planned activities to TÜV
WP 1	AI-safety functional safety management (AI-FSM): IKERLAN is currently working on the adaptation of Ikerlan's SIL 3 FSM to consider new procedures required by AI systems (data management, training, inference). IKERLAN requests TÜV Rheinland to review the documentation (FSM guidelines and templates) and provide feedback and a review report.
WP 2	Railway safety concept: TÜV review and assessment of a safety concept, where AI is used for visual perception tasks of a railway safety function for collision avoidance. *[Quotation]*

- WP0: KoM meeting
- WP1- Activity 1: AI-FSM
- WP2- Activity 2: Railway safety concept

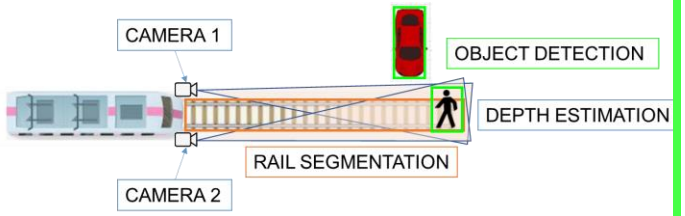
Reference software development process



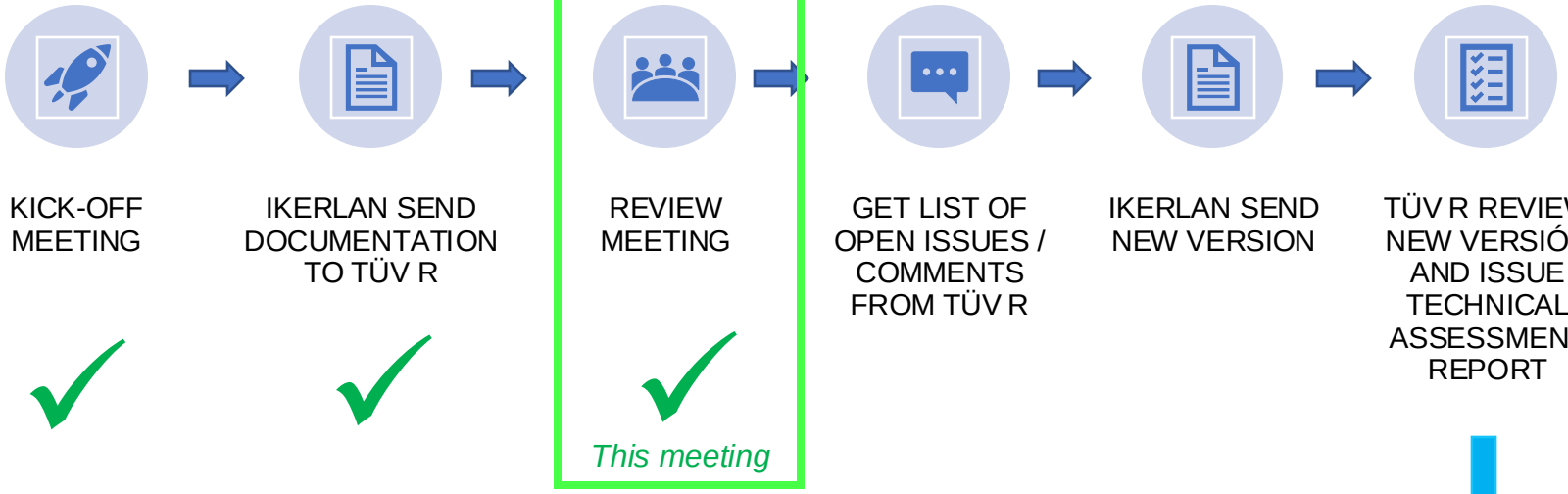
Reference safety architecture



Railway case study

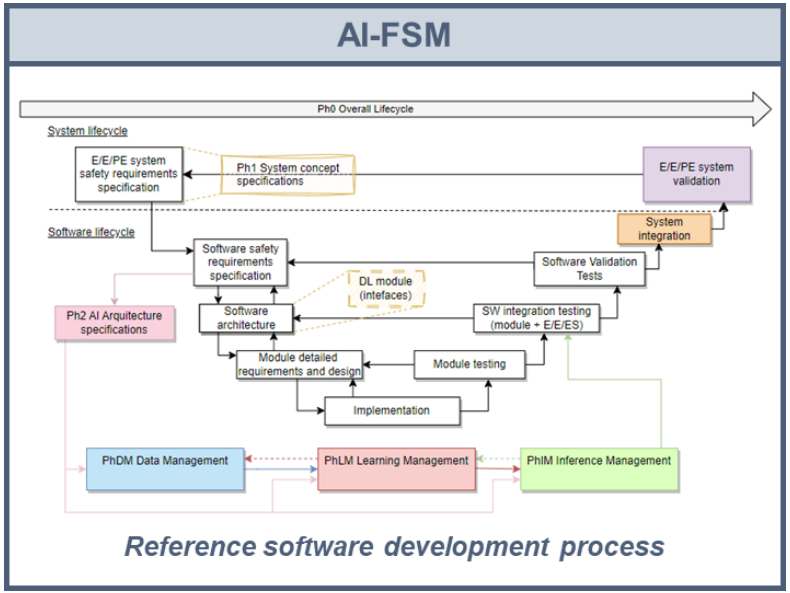


Methodology – per activity



Equivalent to that obtained in previous EU projects (e.g., UP2DATE)

Safety architecture patterns



Need for runtime safety mechanisms to deal with:

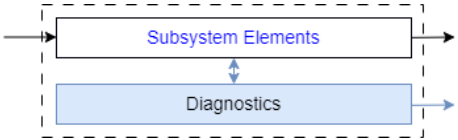
- *Random and residual systematic faults*
- *HW / SW platform complexity: integration problems (e.g., determinism, interferences on mixed-criticality approaches, use of resources…)*
- *DL model insufficiencies*
- *Support DL explainability*
- *…*

Reference safety architecture patterns for the adoption of DL in safety-critical systems with varying safety requirements

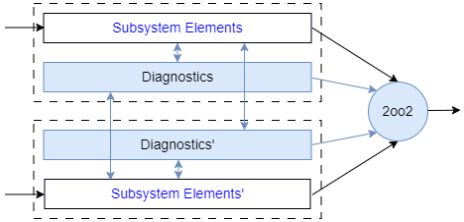
Safety architecture patterns

Safety pattern: Generic solutions for commonly recurring design problems with the aim of simplifying and standardizing the design process

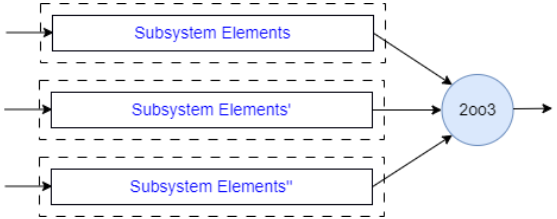
Common examples:



Single channel with diagnostics (1oo1D)

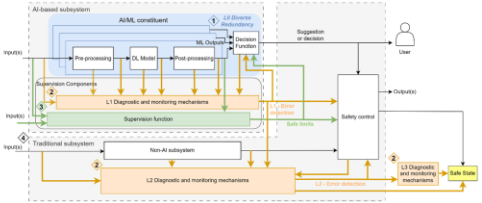


Dual channel with diagnostics (2oo2D)

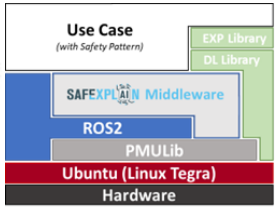


Triple Module Redundancy (TMR) with majority voter (2oo3)

Extend and combine common patterns from traditional Functional Safety (FUSA) to address the new challenges brought by DL-based approaches in complex HW/SW platforms



Reference safety architecture



Safety Pattern 1

Safety Pattern 2

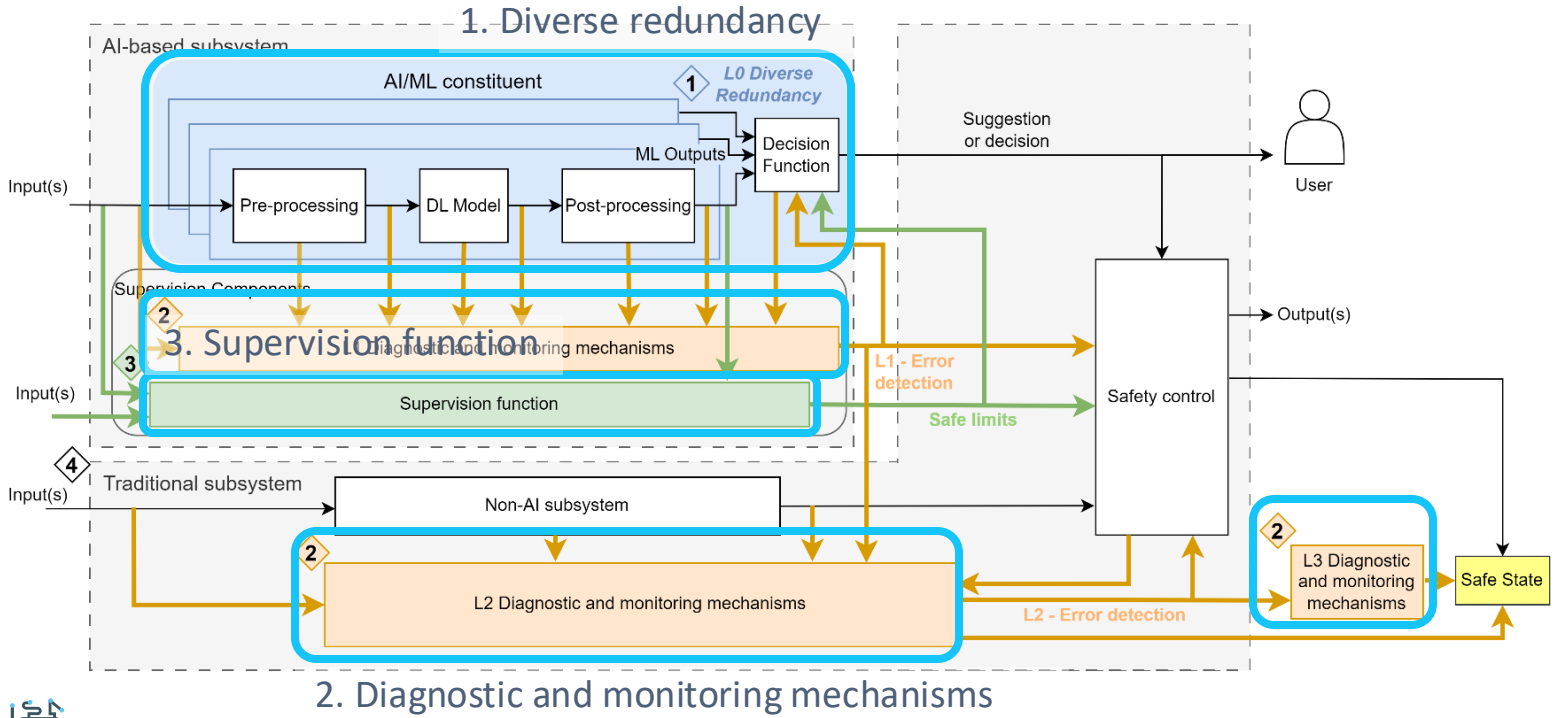
Safety Pattern 3

Evaluation on NVIDIA JETSON AGX ORIN & SAFEXPLAIN SW Stack

SAFEXPLAIN

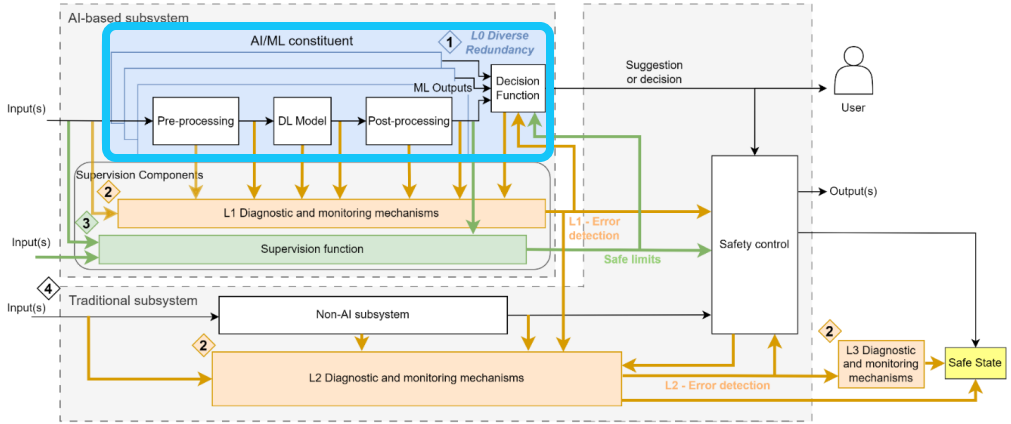
Reference safety architecture

- Reference safety architecture pattern for safe AI-based systems



Reference safety architecture

- L0 Diverse Redundancy



- Inference Platform diversity
  - Inputs (diverse cameras, sensors, input image flips…)
  - Processing resources (accelerators, CPUs…)
  - …
- DL model Development diversity
  - Model Architecture
  - Execution framework (e.g., TF lite, pytorch, darknet…)
  - Hyperparameters
  - Training datasets, process or platform
  - …
- Concept diversity: different problem formulation with same final goal
  - Object detection vs object part detection
  - Object detection vs obstacle free path detection
  - …

5 **Inference Platform diversity**

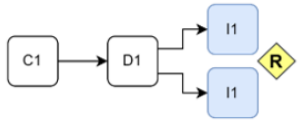
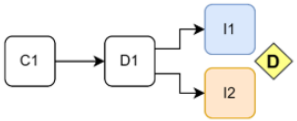
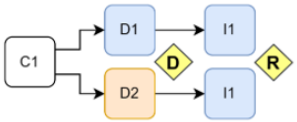
It looks like the diversity is only covering different AI/ML constituents but make use of the same input (sensor) data. Full diversity would mean to also cover the sensor or data input side (sometimes also called dissimilarity). E.g. to use different camera systems or even camera and lidar data inputs for the diverse AI/ML constituents.

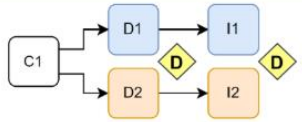
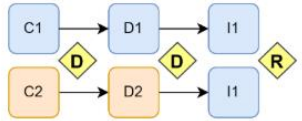
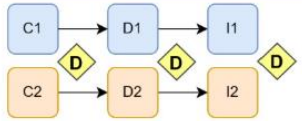
6 **Decision Function**

Remark: To the Decision function there may be the need to assign the highest SIL as it is the point where the diverse redundancy ends. (single point of failure)

Reference safety architecture

- L0 Diverse Redundancy

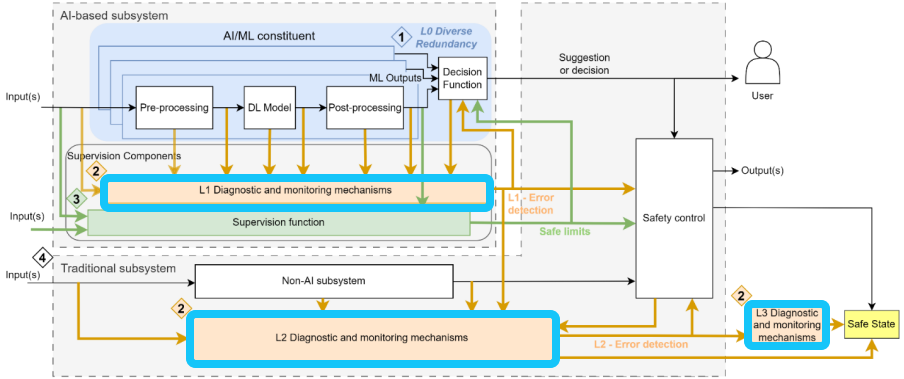
ID	Type of Diverse redundancy: Concept – Development – Inference	Diagnostic Coverage	Description
DRS_1		Low	Single DL model development, replicated in the inference platform without diversity (each replica uses same SW stack and processing elements (e.g., GPU)). Allows addressing traditional FUSA risk factors.
DRS_2		Low to medium	Single DL model development, replicated in the inference platform with diversity (each replica uses different resources (different inputs, processing elements (e.g., GPU and CPU), etc.)). Allows addressing traditional FUSA risk factors.
DRS_3		Low to medium	Development of two different DL models based on same concept (e.g., object detection) with diversity in the model development (training process, training data, model architecture, AI framework, etc.). Inference without diversity (each replica uses same SW stack and processing elements (e.g., GPU)). Allows addressing AI & traditional FUSA risk factors as well as AI performance insufficiencies.

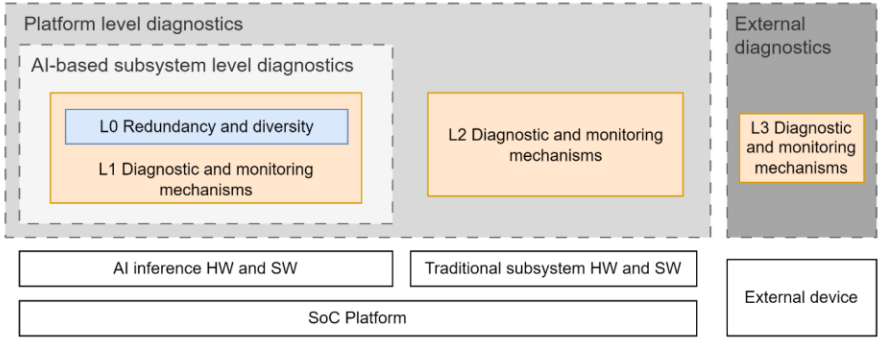
ID	Type of Diverse redundancy: Concept – Development – Inference	Diagnostic Coverage	Description
DRS_4		Medium to high	Combination of DRS_2 and DRS_3 with DL model Development diversity and Inference Platform diversity. Allows addressing AI & traditional FUSA risk factors as well as AI performance insufficiencies.
DRS_5		Medium to high	Development of two different DL models based on different concepts (e.g., one for object detection and the other for obstacle free path detection). This requires DL model Development diversity. Inference without diversity (each replica uses same SW stack and processing elements (e.g., GPU)). Allows addressing AI & traditional FUSA risk factors as well as AI performance insufficiencies.
DRS_6		High	Combination of DRS_4 and DRS_5 with concept diversity, DL model Development diversity and Inference Platform diversity. Allows addressing AI & traditional FUSA risk factors as well as AI performance insufficiencies.

R Redundancy D Diverse Redundancy

Reference safety architecture

- Diagnostic and monitoring mechanisms





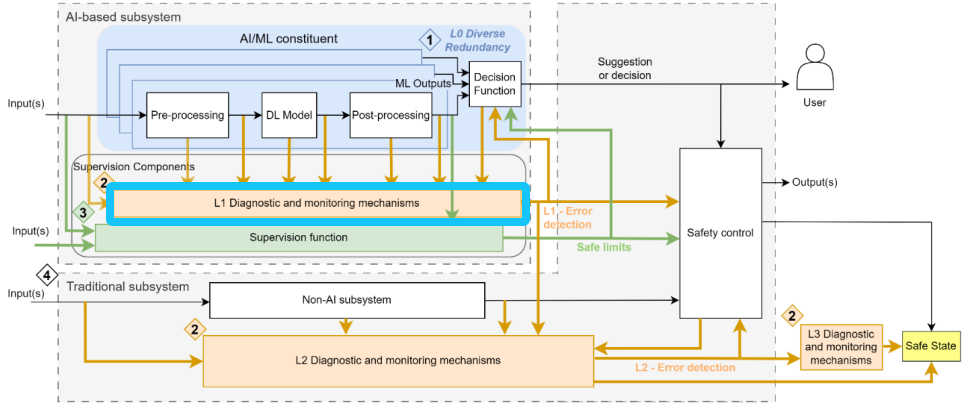
L0 - L1 – AI-based subsystem level diagnostics: runtime errors or model insufficiencies and anomalies on the AI subsystem and the elements required for its execution (e.g., accelerators, AI frameworks, etc.)

L2 – Platform level diagnostics: runtime errors on additional platform HW and SW components following traditional functional safety practices and diagnostics techniques (e.g., memory self-tests, freedom from interference at platform level…)

L3 – External diagnostics: external watchdog or microcontroller

Reference safety architecture

- Diagnostic and monitoring mechanisms – L1DM mechanisms – AI subsystem



L1DM - Inputs

8

Remark: most data Inputs are generated by sensors (cameras, LIDARs etc.), so diagnostics on sensor level could also be integrated here.

L1DM - Model

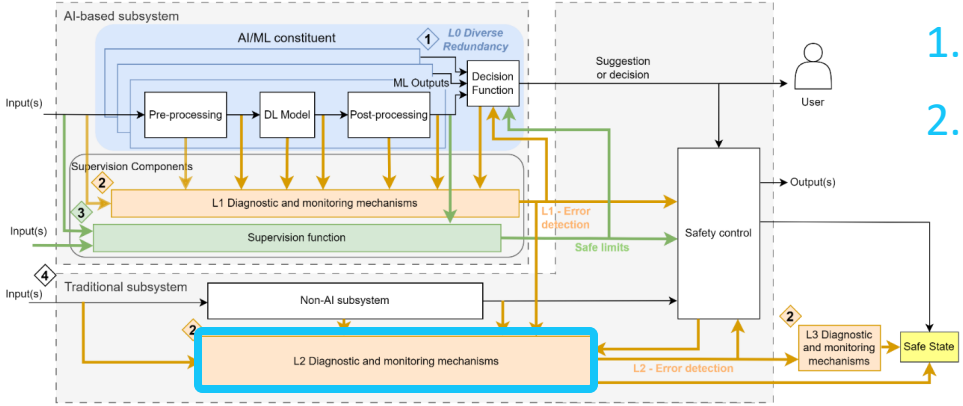
9

Monitoring the neuron activation patterns during operation requires introspection into the Neural Network layers during operation. That may not be possible for all kind of DNN-models without essential negative influence on model performance

1. Inputs: diagnostic mechanisms for input correctness, data quality, data redundancy, temporal consistency…
2. Model: diagnostic and monitoring mechanisms for execution errors, timing, program sequence, neuron activation patterns…
3. Outputs: diagnostic mechanisms for outputs, plausibility checks, input-output correlation, temporal consistency …
4. Resource usage: monitoring mechanisms for resource usage (e.g., CPU/GPU usage, memory usage)

Reference safety architecture

- Diagnostic and monitoring mechanisms – L2DM mechanisms – Traditional subsystem



L2DM – Interference mitigation and control

10

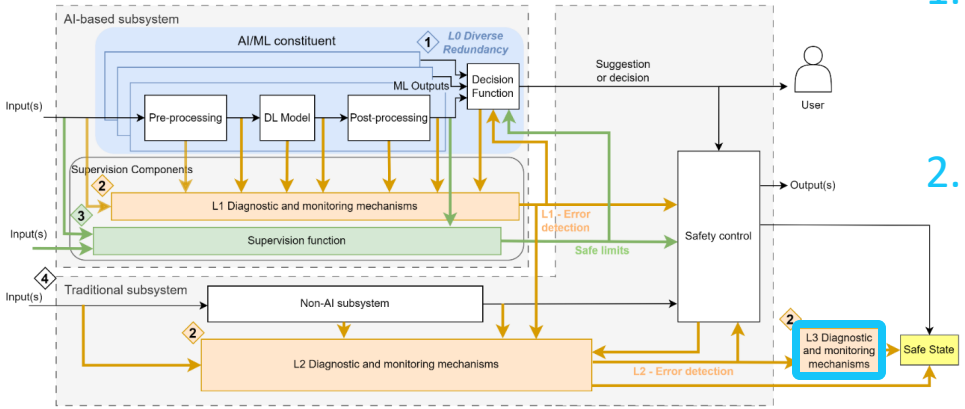
"…verified that the software does not exceed the use of available resources even in worst-case scenarios"

Remark: that should be regarded as the sum-up of all worst-case conditions for all SW-Elements at the same time.

1. Traditional functional safety diagnostics
2. Advanced diagnostic approaches for high-performance platforms
   - L2DM Configuration check – check that the configuration defined and verified during system development is kept at runtime.
   - L2DM Interference mitigation and control – check that all critical platform SW components meet with their deadline and if this is the case, it refreshes an external watchdog (L3).
   - L2DM Health Management – is the responsible of triggering the required reaction whenever L0, L1 or L2 diagnostics and monitoring detects an error

Reference safety architecture

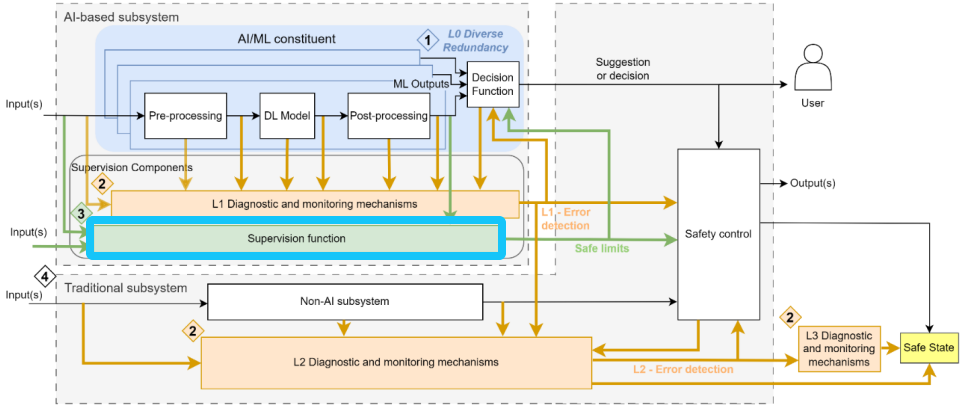
- Diagnostic and monitoring mechanisms – L3 mechanisms – External



1. Traditional external safety mechanisms (e.g., external watchdog, time and power monitoring, external monitoring unit…).
2. Error management for the lower-level diagnostic and monitoring mechanisms (i.e., L0, L1 and L2 mechanisms).

Reference safety architecture

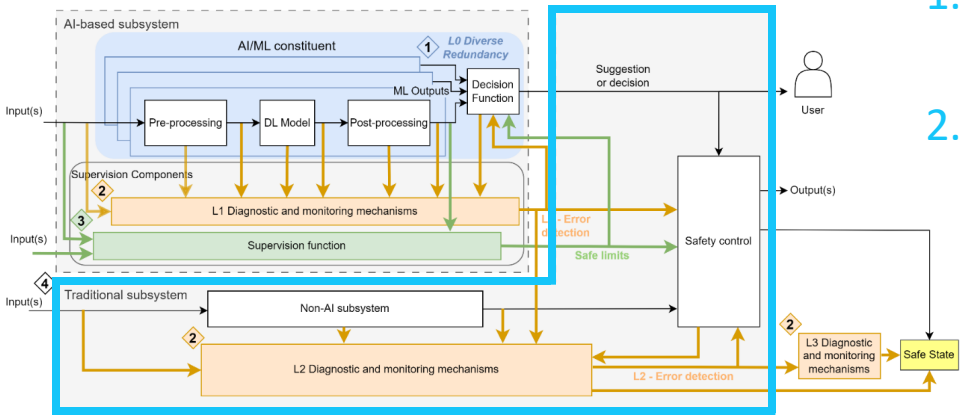
- Supervision function



1. Check the appropriateness of the environment
2. Supervise the output of the ML constituent to identify unsafe situations
3. Stablish the limits for safe operation, providing a safe envelope
4. Provide explanations on the DL model

Reference safety architecture

- Traditional subsystem



1. Functions complementary to the ML constituent.
2. Fallback subsystem – safe back-up function to the ML component that takes over the system if a functional safety problem is detected.

Sources of Risk

- AI Risk Factors

Risk factor	AI technology element	Source
Traditional FUSA risk factors (systematic / random)	All AI technology elements (i.e., computational hardware, OS / middleware, libraries, ML framework and ML model).	IEC 61508 and derived FUSA standards (ISO 26262, EN 5012x, …)
HPEC platform integration risk factors	Computational hardware and lower-level SW (i.e., OS / middleware).	CAST-32A / AMC-20-193 ISO 26262 part 11
AI performance insufficiency	ML model or framework.	ISO/DIS 21448 (SOTIF)
AI & FUSA risk factors (low/medium integrity level)	All AI technology elements (i.e., computational hardware, OS / middleware, libraries, ML framework and ML model).	AI and FUSA Technical Reports (ISO/IEC TR 5469)

3 **AI Risk Factors**

Why clause 8.4.3 "issues related to learning from environment" of ISO/IEC TR 5469 is not explicitly regarded as one of the relevant essential application dependent properties?

1 **AI Risk Factors**

It is kind of pseudo stochastic - as it's not fully comparable with the real stochastic behavior of HW failures.

2 **AI Risk Factors**

In classical FuSa there are only random faults for HW components. In ML/AI based systems effects may occur that seem to be "random" but they are not really random; only the number of possible variations of parameters is extremely high and therefore they are seen and handled as pseudo randomised distributions. The underlying internal causes of faulty behaviour are still systematic - but the systematic is not easily detectable.

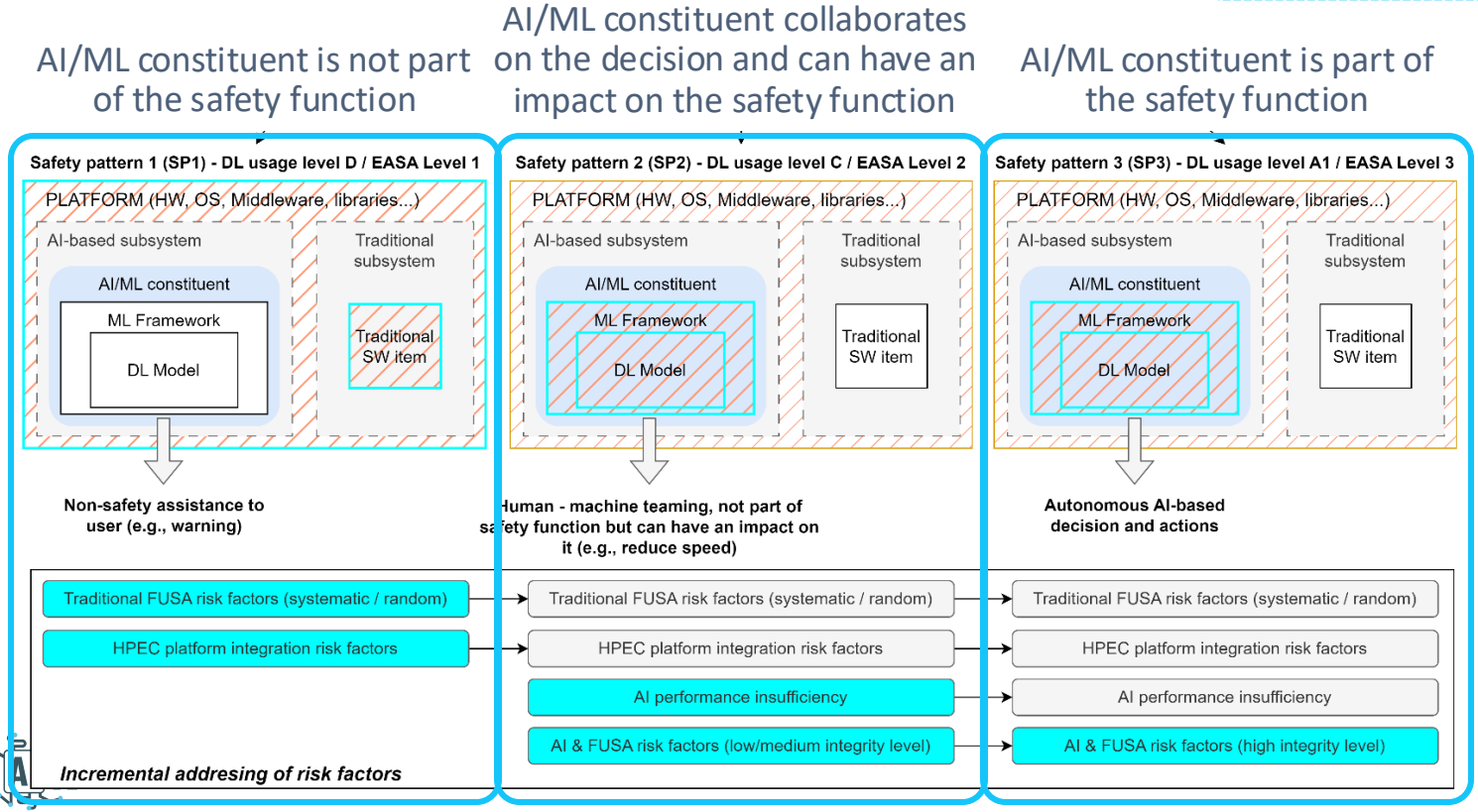
Incremental Strategy

Incremental Strategy

4

What is the meaning of the different hatching of areas?

- Incremental strategy for AI adoption in safety critical systems



Automatic Train Operation (ATO) system

- Detect obstacles in the railway and estimates distance by assistance of cameras. According to the detected obstacle (i.e., critical / non-critical), the distance to it and its location (in / outside the track), the system responds accordingly (i.e., warn the driver, activate service brake, activate emergency brake).

12 ATO System

Question: Only cameras? Why not additional Sensor technology: e.g. IR-camera, or LIDAR? Most perception systems rely on more than just one set of cameras. (except for Tesla, but there have been incidents based on the lack of different sensors)

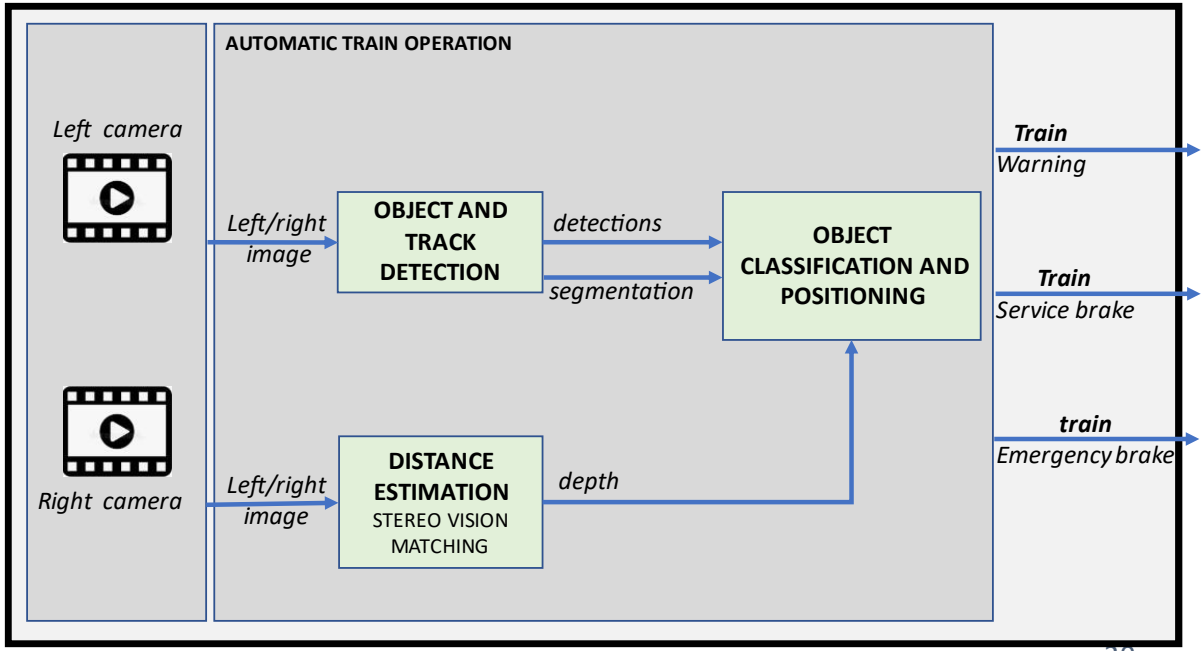
13 ATO System

"(i.e., warn the driver, activate service brake, activate emergency brake)". or sound the horn in order to warn persons or animals in or close to the track

14 ATO System

"objects outside the track"

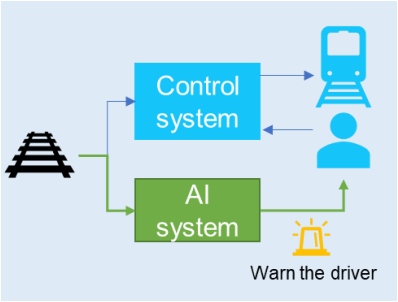
How far outside of the track? - At least the clearance profile of the train on the predicted ego Track should be applied here



Automatic Train Operation (ATO) system

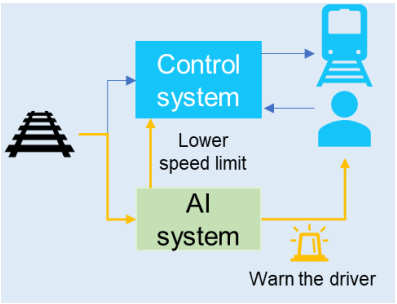
- Detect obstacles in the railway and estimates distance by assistance of cameras. According to the detected obstacle (i.e., critical / non-critical), the distance to it and its location (in / outside the track), the system responds accordingly (i.e., warn the driver, activate service brake, activate emergency brake).

GoA 1 – SP1



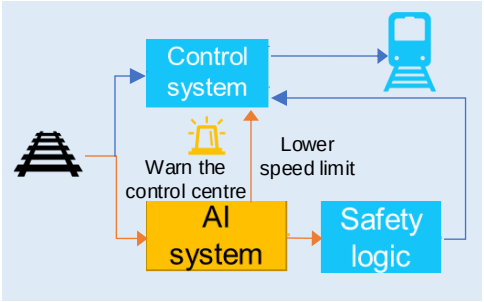
The **AI system detects an obstacle** (another train, person, car…) in the tracks and **warns the driver** of the potential danger. The **driver** must act safely activating the **brakes** if necessary.

GoA 2 – SP2



The **AI system detects an obstacle** in line of collision and **warns the driver**. The AI system **activates the service brake (SIL2)** reducing the speed of the train. The **driver** can **activate the emergency brake (SIL4), or** temporarily **deactivate the service brake (SIL2)**.

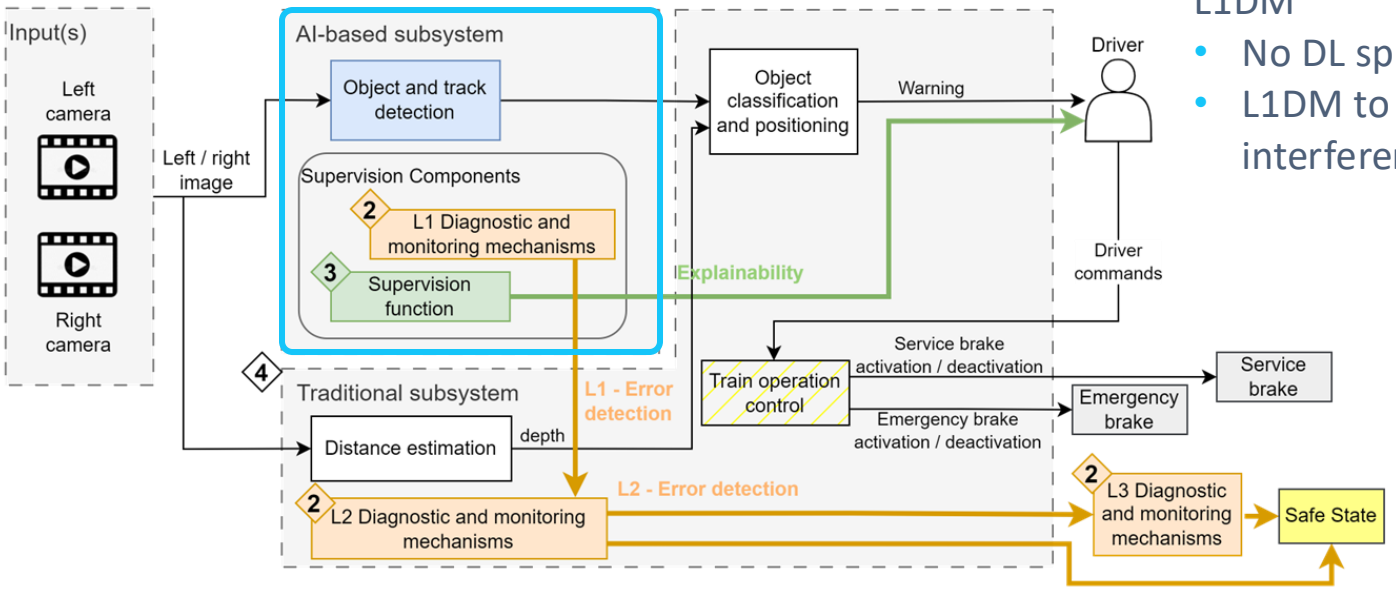
GoA 3 or 4 – SP3



The **AI system detects an obstacle** in line of collision and **warns the control center**. The AI system **activates the service brake (SIL2)** and **calculates the braking distance**, **activating the emergency brake (SIL4)** if the collision distance is below a threshold.

ATO System (GoA 1)

- Safety Pattern 1 (SP1) – Diagnostic and monitoring mechanisms

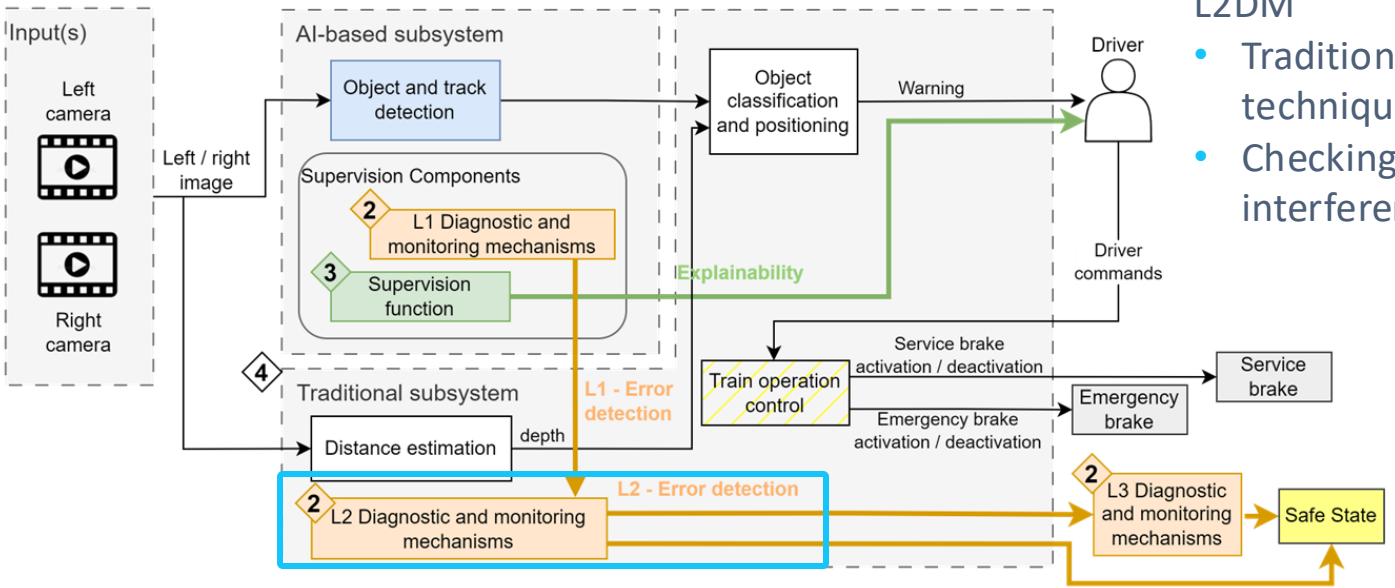


L1DM

- No DL specific diagnostics
- L1DM to guarantee freedom from interference

ATO System (GoA 1)

- Safety Pattern 1 (SP1) – Diagnostic and monitoring mechanisms

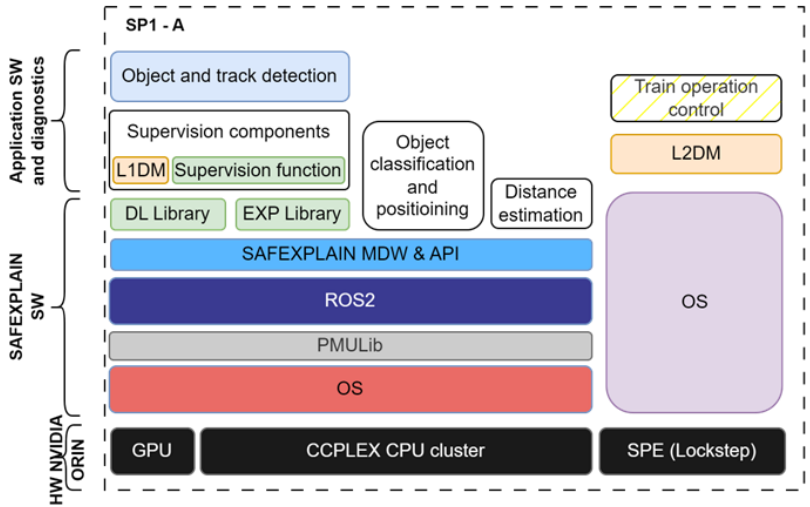


L2DM

- Traditional functional safety techniques and measures
- Checking system configuration for interference mitigation and control

ATO System (GoA 1)

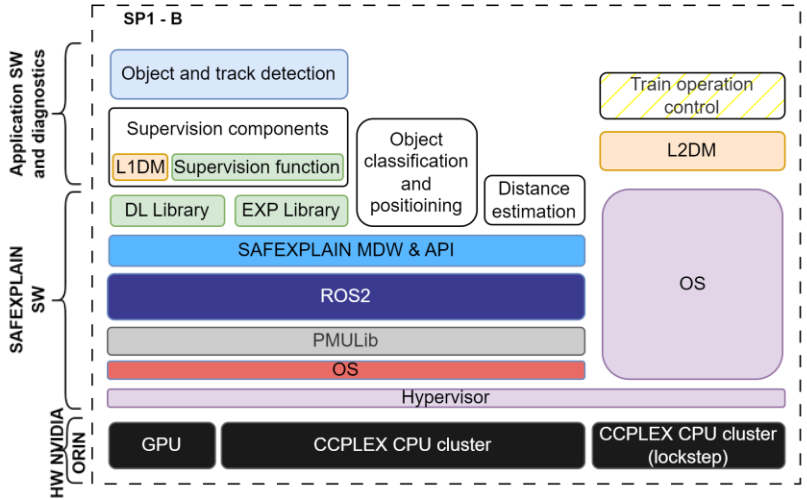
- SP1 to NVIDIA Orin resource allocation and configuration option A



SP1 Element	Safety / non-safety	SP1 - A NVIDIA Orin resources and configuration
Object and tract detection	Non-safety AI based SW	CCPLEX CPU Cluster (Cortex A78) GPU for AI inference Memory controller fabric and traffic from CPU cluster to GPU MMUs for spatial independence SAFEXPLAIN SW Stack
Supervision components	Non-safety traditional or AI based SW	
Distance estimation	Non-safety traditional SW	
Object classification and positioning	Non-safety traditional SW	
Train operation control (safety function)	Safety traditional SW	SPE in lockstep configuration No sharing of caches with CCPLEX MMUs for spatial independence Safe RTOS on top of SPEs
L2DM	Safety traditional SW	
Pros		Highest independence: different processing elements, less shared resources, different OS and SW stack.
Cons		Limited performance on SPEs.

ATO System (GoA 1)

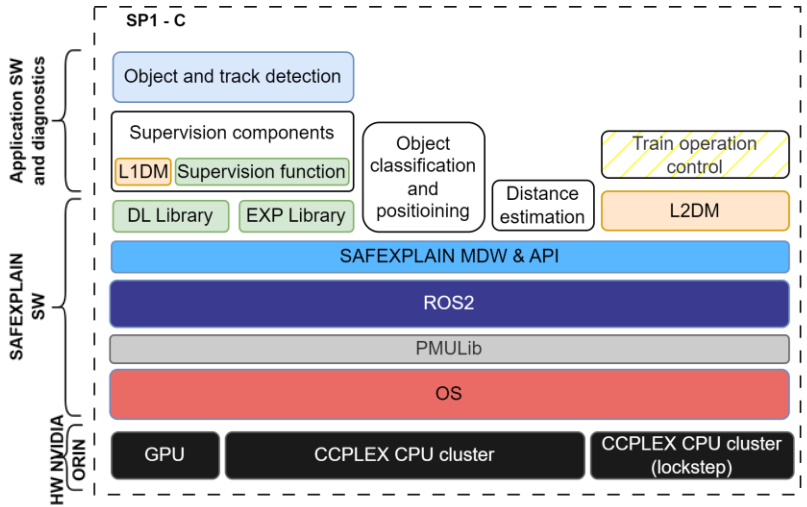
- SP1 to NVIDIA Orin resource allocation and configuration option B



SP1 Element	Safety / non-safety	SP1 - B NVIDIA Orin resources and configuration
Object and tract detection	Non-safety AI based SW	CCPLEX CPU Cluster (Cortex A78) – different CPU cluster for safety SW GPU for AI inference Memory controller fabric and traffic from CPU cluster to GPU MMUs for spatial independence L4 cache partitioning or disabled SAFEXPLAIN SW Stack separated from safety SW by hypervisor
Supervision components	Non-safety traditional or AI based SW	
Distance estimation	Non-safety traditional SW	
Object classification and positioning	Non-safety traditional SW	
Train operation control (safety function)	Safety traditional SW	CCPLEX CPU Cluster (Cortex A78) in lockstep – different CPU cluster for safety SW MMUs for spatial independence L4 cache partitioning or disabled Safe RTOS on top of hypervisor
L2DM	Safety traditional SW	
Pros		Better performance while preserving independence: different CPU clusters with partitioning approaches, different OS and SW stack.
Cons		Need of a hypervisor.

ATO System (GoA 1)

- SP1 to NVIDIA Orin resource allocation and configuration option C



SP1 Element	Safety / non-safety	SP1 - C NVIDIA Orin resources and configuration
Object and tract detection	Non-safety AI based SW	CCPLEX CPU Cluster (Cortex A78) – different CPU cluster for safety SW GPU for AI inference Memory controller fabric and traffic from CPU cluster to GPU MMUs for spatial independence L4 cache partitioning or disabled SAFEXPLAIN SW Stack
Supervision components	Non-safety traditional or AI based SW	
Distance estimation	Non-safety traditional SW	
Object classification and positioning	Non-safety traditional SW	
Train operation control (safety function)	Safety traditional SW	CCPLEX CPU Cluster (Cortex A78) in lockstep – different CPU cluster for safety SW MMUs for spatial independence L4 cache partitioning or disabled SAFEXPLAIN SW Stack
L2DM	Safety traditional SW	
Pros		Ease of integration, same SW stack for all platform elements.
Cons		Less independence, required safety guarantees on the SAFEXPLAIN SW Stack.

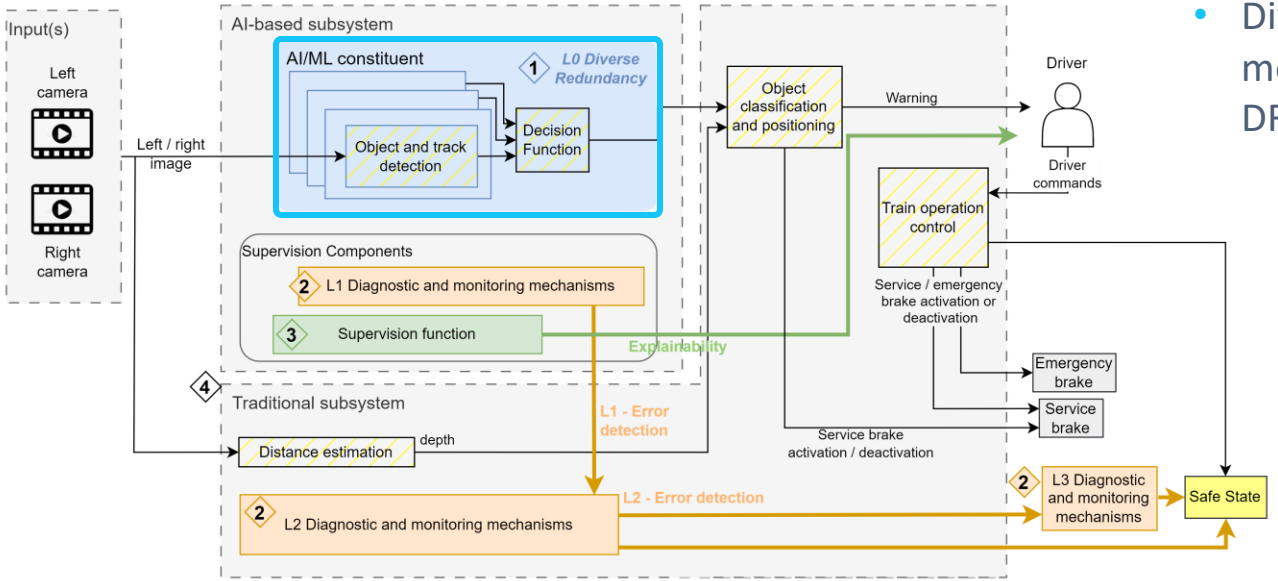
SAFEXPLAIN SW Stack

16

"Required safety guarantees on the SAFEXPLAIN SW Stack." This may be required for the full Stack (OS, Libs, ROS2, MDW & API), as all of it is shared between the non-safety and the safety parts

ATO System (GoA 2)

- Safety Pattern 2 (SP2) – L0 Diverse redundancy



- Diverse redundancy schemes with medium or high DC (DRS_4, DRS_5 or DRS_6)
  - Diverse processing resources (GPU and CPU, GPU and other AI accelerators)
  - Diverse model development (framework, model architecture…)
  - Concept diversity (object detection vs. Object part detection

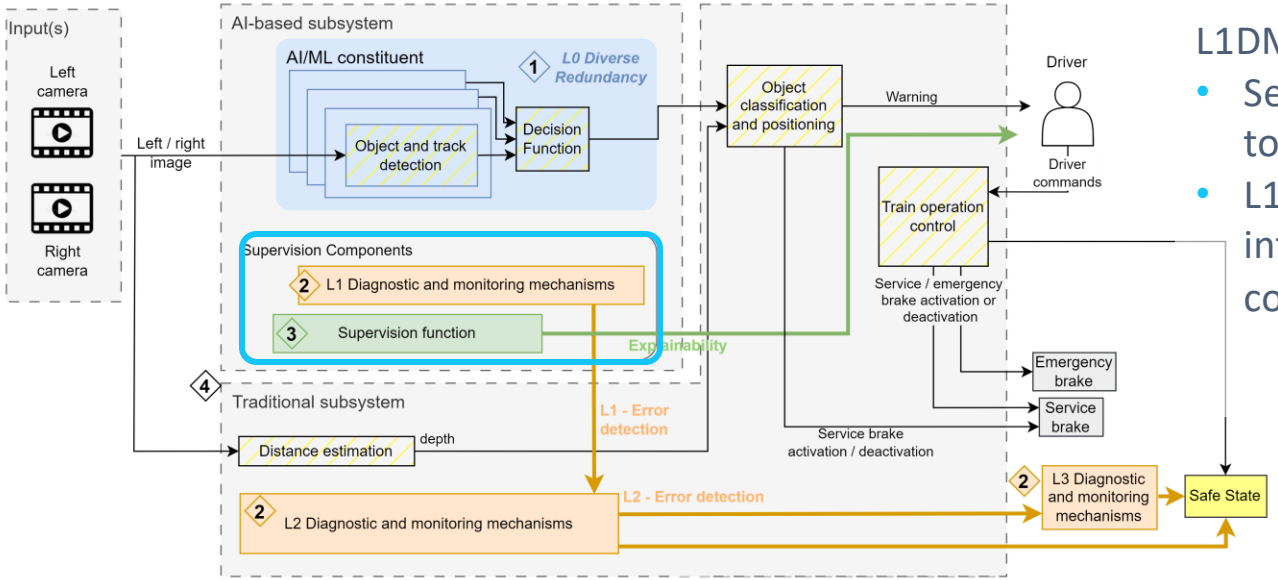
SP2 – L0 Diverse Redundancy

17

"Whereas the focus or SP2 is on HFT = 0 with low to medium diagnostic coverage." But then only SIL1 can be achieved as maximum according to table 6!! The required SIL2 can only be achieved by medium to high DC (>90%)!!

ATO System (GoA 2)

- Safety Pattern 2 (SP2) – Diagnostic and monitoring mechanisms

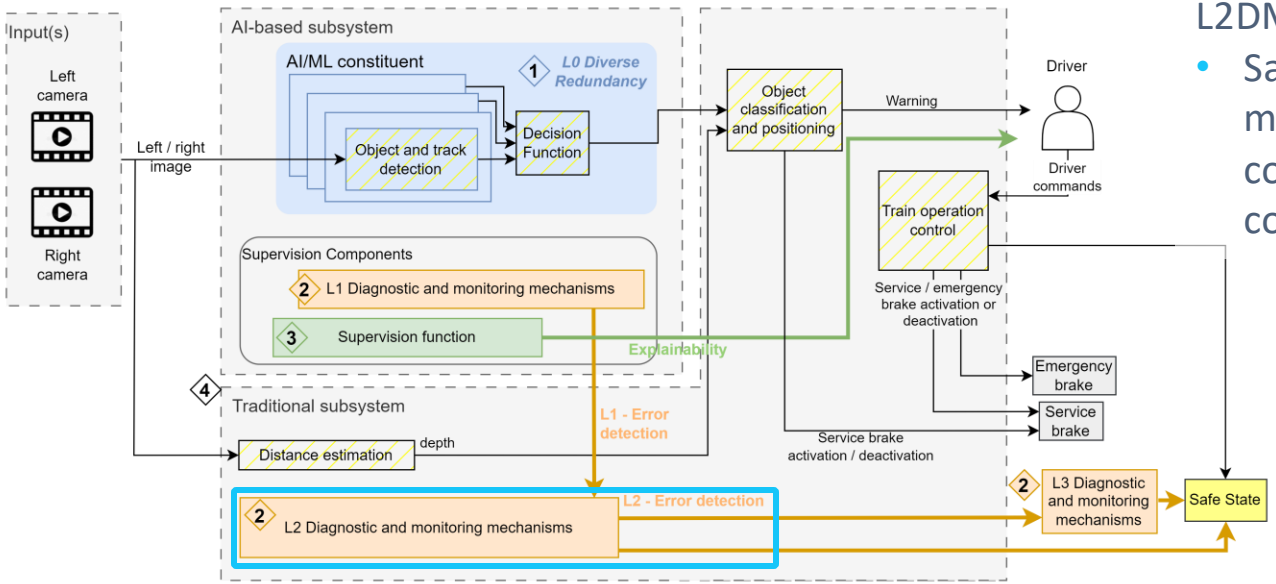


L1DM

- Select L1DM diagnostics mechanisms to reach a high diagnostic coverage
- L1DM to guarantee freedom from interference, also in the AI/ML constituent

ATO System (GoA 2)

- Safety Pattern 2 (SP2) – Diagnostic and monitoring mechanisms

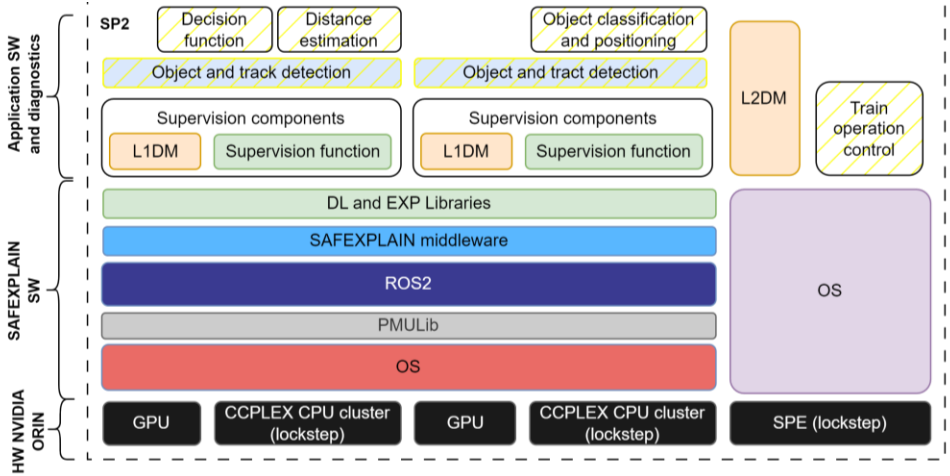


L2DM

- Same as SP1 L2DM (traditional mechanisms + checking system configuration), but also for the AI/ML constituent

ATO System (GoA 2)

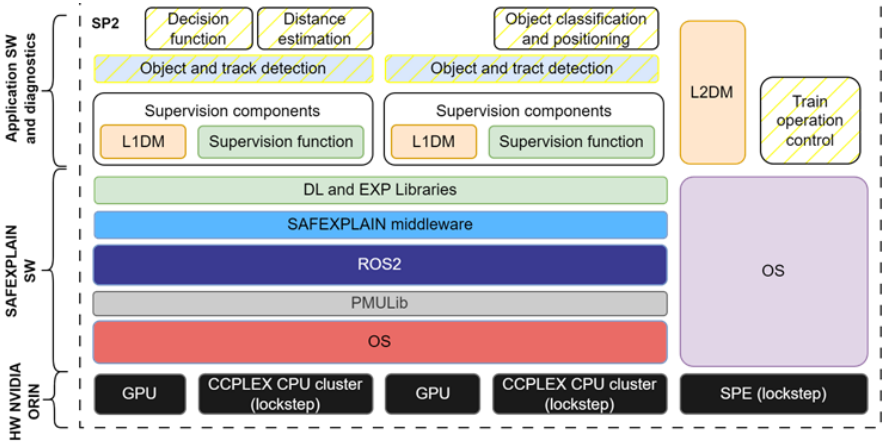
- SP2 to NVIDIA Orin resource allocation and configuration option



SP2 Element	Safety / non-safety	SP2 - A NVIDIA Orin resources and configuration
Object and track detection	AI based SIL 2 SW	Two instances, each in one separate CCPLEX CPU Cluster (Cortex A78) in lockstep configuration GPU for AI inference (depending on the DRS CPU or other computing resources could also be used to improve diversity) Memory controller fabric and traffic from CPU cluster to GPU MMUs for spatial independence SAFEXPLAIN SW Stack
Supervision components	Traditional or AI based SIL 2 SW	Each AI/ML constituent has each own L1DM and optionally each own supervisor function (depends on user application). Depending on the implementation of the supervision component, it may need GPUs for improved performance (e.g., AI based supervision function). The supervision components can share same CCPLEX CPU Cluster (Cortex A78) in lockstep configuration as the AI/ML constituent. MMUs for spatial independence SAFEXPLAIN SW Stack

ATO System (GoA 2)

- SP2 to NVIDIA Orin resource allocation and configuration option



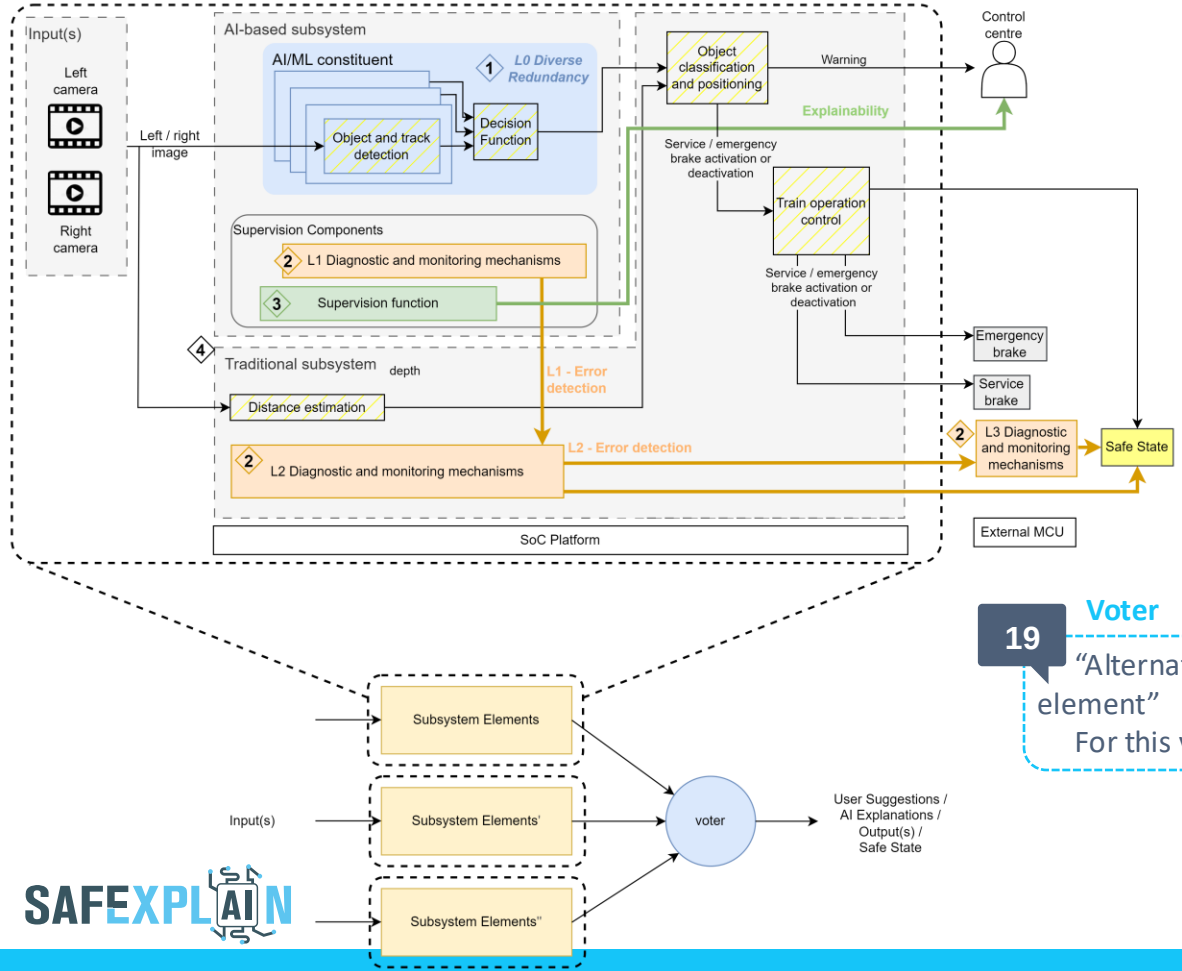
SP2 Element	Safety / non-safety	SP2 - A NVIDIA Orin resources and configuration
Decision function	Traditional SIL 2 SW	These SW components can run on any of the CCPLEX CPU Cluster (Cortex A78) in lockstep configuration used for the AI/ML constituent with the same configuration, since they have the same integrity level.
Distance estimation		
Object classification and positioning		
Train operation control	Traditional SIL 4 SW	CCPLEX CPU Cluster (Cortex A78) or SPE in lockstep configuration. MMUs for spatial independence L4 cache partitioning or disabled SAFEXPLAIN SW Stack or different OS on top of SPEs or hypervisor
L2DM	Traditional SIL 4 SW	

18

Decision Function

As the decision function is essential for bringing together the redundant paths it may have a higher integrity requirement as each of the AI based SIL2 SW components it gets the input from

ATO System (GoA 3 o 4)



- Safety Pattern 3 (SP3)
  - Same as SP2 with off-chip redundancy

we assume that for SP3 an HFT > 0 is required

19 **Voter**

"Alternatively, the voting can be done in an external element"
For this voter then also SIL4 needs to be applied

Next steps

Railway SC – Activity planning

WP No.	Work Package (WP)
Activity 2: Review and Assessment of Safety concept of the safe Railway Use case based on AI collision avoidance system w.r.t IEC 61508 / EN 5012x and ISO 5469 drafts	
WP 2.1	**Review** of provided draft Safety Concept Document, Compilation of a list of comments and open issues
WP 2.2	Review **Workshop** in Arrasate-Mondragón (One day, two experts of TÜV Rheinland) including preparation and follow-up
WP 2.3	Issue of **Technical Note with open issues and comments** of the Review of the draft Safety Concept Document
WP 2.4	**Assessment** of revised Safety Concept Document, Compilation of final Assessment Report

KICK-OFF MEETING ✓

IKERLAN SEND RAILWAY SC TO TÜV R ✓

REVIEW MEETING ✓ *This meeting*

GET LIST OF OPEN ISSUES / COMMENTS FROM TÜV R *Early Feb. 25*

IKERLAN SEND NEW VERSIÓN OF RAILWAY SC *March 25*

TÜV R REVIEW NEW VERSIÓN AND ISSUE TECHNICAL ASSESSMENT REPORT

SAFEXPLAIN

THANK YOU!

SAFEXPLAIN

Safe and Explainable
Critical Embedded Systems based on AI

Follow us on social media:

www.safexplain.eu

Funded by the European Union

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

Project Consortium

- **BARCELONA SUPERCOMPUTING CENTER (BSC)**
  - https://www.bsc.es/
- IKERLAN, S. Coop (IKR)
  - https://www.ikerlan.es/
- AIKO SRL (AIKO)
  - https://www.aikospace.com/
- RISE RESEARCH INSTITUTES OF SWEDEN AB (RISE)
  - https://www.ri.se/
- NAVINFO EUROPE BV (NAV)
  - https://www.navinfo.eu/
- EXIDA DEVELOPMENT SRL (EXI)
  - https://www.exida-eu.com/

SafeYOLO

Contextualization

AI-based applications usually prioritize performance, often leaving safety aside

Our objective is to avoid systematic errors and detect and mitigate runtime errors

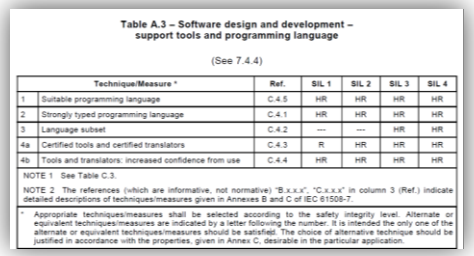


Table A.3 – Software design and development – support tools and programming language

(See 7.4.4)

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Suitable programming language	C.4.5	HR	HR	HR	HR
2	Strongly typed programming language	C.4.1	HR	HR	HR	HR
3	Language subset	C.4.2	---	---	HR	HR
4a	Certified tools and certified translators	C.4.3	R	HR	HR	HR
4b	Tools and translators: increased confidence from use	C.4.4	HR	HR	HR	HR

NOTE 1 See Table C.3.

NOTE 2 The references (which are informative, not normative) "B.x.x.x", "C.x.x.x" in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.

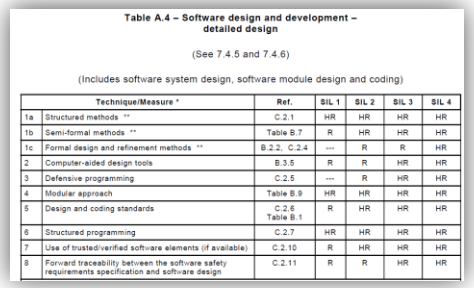


Table A.4 – Software design and development – detailed design

(See 7.4.5 and 7.4.6)

(Includes software system design, software module design and coding)

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1a	Structured methods **	C.2.1	HR	HR	HR	HR
1b	Semi-formal methods **	Table B.7	R	HR	HR	HR
1c	Formal design and refinement methods **	B.2.2, C.2.4	---	R	R	HR
2	Computer-aided design tools	B.3.5	R	R	HR	HR
3	Defensive programming	C.2.5	---	R	HR	HR
4	Modular approach	Table B.9	HR	HR	HR	HR
5	Design and coding standards	C.2.6 Table B.1	R	HR	HR	HR
6	Structured programming	C.2.7	HR	HR	HR	HR
7	Use of trusted/verified software elements (if available)	C.2.10	R	HR	HR	HR
8	Forward traceability between the software safety requirements specification and software design	C.2.11	R	R	HR	HR

C-based Implementation

MISRA C coding guideline

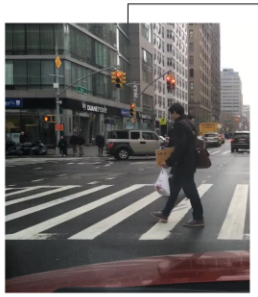
C-based object detector accomplishing with MISRA C coding guidelines

- *Drawback: Execution time*
- *Solution: "SafeYOLO", a safety mechanism to periodically diagnose accelerator-based implementations*

Set-up

- Input: images
- Version: YOLOv4 (link) employing a specific layer configuration extracted from yolo_v7
- Framework: C-based Darknet
- We have focused on inference phase leaving aside training phase
- MISRA C compliant analysis based on Polyspace:
  - Polyspace Bug Finder tool
  - MISRA C:2012

Main operations performed in darknet-based YOLO



Network configuration

- Load Alphabet
- Network configuration
- Layer Configuration
- ...
- Activation weights

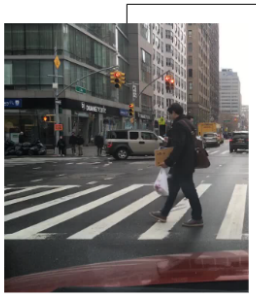
Inference

Conv
Conv
C2f
Conv
C2f
Conv
C2f
Conv
C2f
SPPF
C2f
Concat
Upsample
C2f
Concat
Upsample
Conv
Concat
C2f
Conv
Concat
C2f
Detect
Detect
Detect

Postprocessing

- Get network boxes
- Sort Predictions
- ...
- Draw predictions

Main operations performed in darknet-based YOLO



Network configuration

- Load Alphabet
- Network configuration
- Layer Configuration
- ...
- Activation weights

Inference

Conv
Conv
C2f
Conv
C2f
Conv
C2f
Conv
C2f
SPPF
Upsample
Concat
C2f
Upsample
Concat
C2f
Conv
Concat
C2f
Detect
Conv
Concat
C2f
Detect
Detect

Postprocessing

- Get network boxes
- Sort Predictions
- ...
- Draw predictions

Done

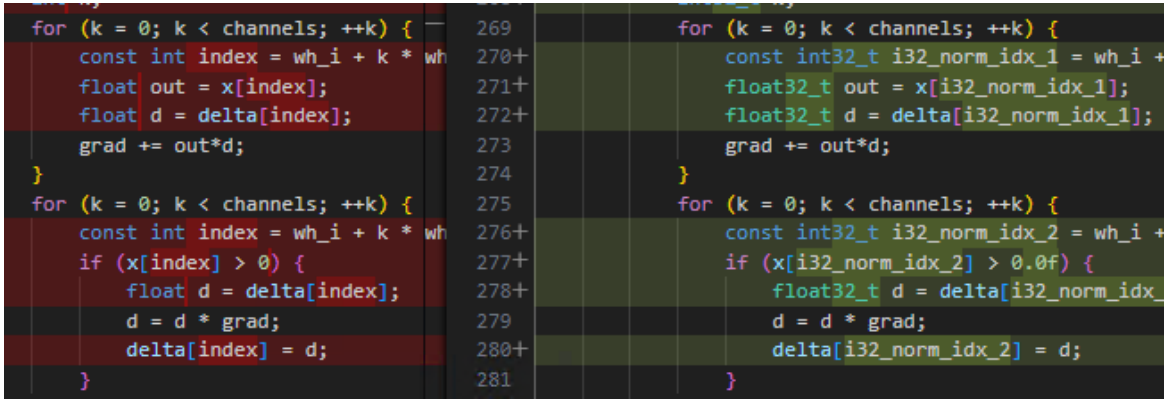
MISRA-C Violations

MISRA C:2012	31378
Dir 1 The implementation	2071
Dir 4 Code design	10562
2 Unused code	227
3 Comments	67
4 Character sets and lexical conventions	5
5 Identifiers	1262
7 Literals and constants	928
8 Declarations and definitions	1440
9 Initialization	40
10 The essential type model	4735
11 Pointer type conversions	994
12 Expressions	3375
13 Side effects	112
14 Control statement expressions	837
15 Control flow	1814
16 Switch statements	112
17 Functions	1464
18 Pointers and arrays	319
20 Preprocessing directives	13
21 Standard libraries	876
22 Resources	125

Representative examples:

5. Identifiers:

- Identifiers that define objects or functions with external linkage shall be unique.
- A tag name shall be a unique identifier.
- An identifier declared in an inner scope shall not hide an identifier declared in an outer scope.



SAFEXPLAIN

MISRA-C Violations

MISRA C:2012	31378
Dir 1 The implementation	2071
Dir 4 Code design	10562
2 Unused code	227
3 Comments	67
4 Character sets and lexical conventions	5
5 Identifiers	1262
7 Literals and constants	928
8 Declarations and definitions	1440
9 Initialization	40
10 The essential type model	4735
11 Pointer type conversions	994
12 Expressions	3375
13 Side effects	112
14 Control statement expressions	837
15 Control flow	1814
16 Switch statements	112
17 Functions	1464
18 Pointers and arrays	319
20 Preprocessing directives	13
21 Standard libraries	876
22 Resources	125

Representative examples:

10. The essential type model

- Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category.

```
static float hard_mish_yashas_grad(float x)
{
    if (x > 0)
        return 1;
    if (x > -2)
        return x + 1;
    return 0;
}
```

```
static float32_t hard_mish_yashas_grad(float32_t x)
{
    float32_t f32_grad_aux;       You, 5 months ago •
    if      (x > 0.0f)      {f32_grad_aux = 1.0f;  }
    else if (x > -2.0f)     {f32_grad_aux = x+1.0f;}
    else                    {f32_grad_aux = 0.0f;  }

    return f32_grad_aux;
```

- The value of an expression shall not be assigned to an object with a narrower essential type or of a different essential type category

```
float32_t dist_array(float32_t *a, float32_t *b
{
    int32_t i;
    float32_t sum = 0;
    for(i = 0; i < n; i += sub) {sum += pow(a[i
    return sqrt(sum);
}
```

```
660  float32_t dist_array(float32_t *a, float32_t *b, int32_t n, int32_t sub)
661  {
662      int32_t i;
663      float32_t sum = 0;
664+     for(i = 0; i < n; i += sub) {sum += ((float32_t) pow(a[i]-b[i], 2));}
665+     return ((float32_t)sqrt(sum));
666  }
```

MISRA-C Violations

MISRA C:2012	31378
Dir 1 The implementation	2071
Dir 4 Code design	10562
2 Unused code	227
3 Comments	67
4 Character sets and lexical conventions	5
5 Identifiers	1262
7 Literals and constants	928
8 Declarations and definitions	1440
9 Initialization	40
10 The essential type model	4735
11 Pointer type conversions	994
12 Expressions	3375
13 Side effects	112
14 Control statement expressions	837
15 Control flow	1814
16 Switch statements	112
17 Functions	1464
18 Pointers and arrays	319
20 Preprocessing directives	13
21 Standard libraries	876
22 Resources	125

Representative examples:

14. Control statement expressions

- The controlling expression of an if statement and the controlling expression of an iteration-statement shall have essentially Boolean type.

```
if(params.net.adam){          201+  if(0 != params.net.adam){
if(!(h && w && c)) {          124+  if(0 == ((0 != h) && (0 != w) && (0 != c)))
while(n){                     97+   while(NULL != n){
```

15. Control Flow

- All if … else if constructs shall be terminated with an else statement.
- The body of an iteration-statement or a selection-statement shall be a compound-statement.

```
if (argc > 4) {    //voc.names yolo-voc.cfg
    names_file = argv[1];
    cfg_file = argv[2];
    weights_file = argv[3];
    filename = argv[4];
}
else if (argc > 1) filename = argv[1];
```

```
73   if (argc > 4) {    //voc.names yolo-voc.cfg yolo-voc.weights test.mp4
74       names_file = argv[1];
75       cfg_file = argv[2];
76       weights_file = argv[3];
77       filename = argv[4];
78   }
79   else if (argc > 1) filename = argv[1];        Javier García González, 8 mon
80+  else {/* Intentionally left empty; no action needed for default case */}
81
```

```
561-    if (a < min) return min;
562-    if (a > max) return max;
```

```
566+    if (a < min) {return min;}
567+    if (a > max) {return max;}
```

```
int32_t alphanum_to_int(char c)
{
    return (c < 58) ? c - 48 : c-87;
}
```

```
164  int32_t alphanum_to_int(char c)
165  {
166+     return (c < 58) ? (c - 48) : (c-87);
167  }
```

SAFEXPLAIN

MISRA-C Violations

MISRA C:2012	31378
Dir 1 The implementation	2071
Dir 4 Code design	10562
2 Unused code	227
3 Comments	67
4 Character sets and lexical conventions	5
5 Identifiers	1262
7 Literals and constants	928
8 Declarations and definitions	1440
9 Initialization	40
10 The essential type model	4735
11 Pointer type conversions	994
12 Expressions	3375
13 Side effects	112
14 Control statement expressions	837
15 Control flow	1814
16 Switch statements	112
17 Functions	1464
18 Pointers and arrays	319
20 Preprocessing directives	13
21 Standard libraries	876
22 Resources	125

Representative examples:

16. Switch statements

- Every switch statement shall have a default label.
- An unconditional break statement shall terminate every switch-clause.
- All switch statements shall be well-formed (An unconditional break statement shall terminate every switch-clause + Default clause shall contain statements or comments).

```
77- float activate(float x, ACTIVATION a)
78  {
79      switch(a){
80          case LINEAR:
81-             return linear_activate(x);
82          case LOGISTIC:
83-             return logistic_activate(x);
```

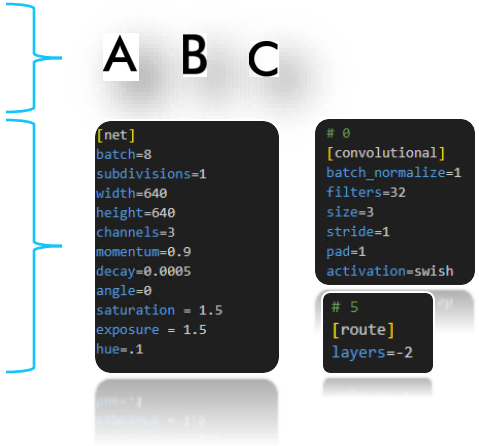
```
39+ float32_t activate(float32_t x, ACTIVATION a)
40  {
41+     float32_t f32_aux_activation_val;
42      switch(a){
43          case LINEAR:
44+             f32_aux_activation_val = linear_activate(x);
45+             break;
46          case LOGISTIC:
47+             f32_aux_activation_val = logistic_activate(x);
48+             break;
```

```
109         case LHTAN:
110-            return lhtan_activate(x);
111-    }
112-    return 0;
```

```
86          case LHTAN:
87+             f32_aux_activation_val = lhtan_activate(x);
88+             break;
89+         default:
90+             f32_aux_activation_val = x;
91+             break;
92+     }
93+     return f32_aux_activation_val;
```

2nd Stage: Main operations performed in darknet-based YOLOv7

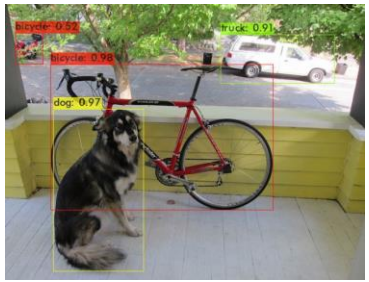
1. Load alphabet (images employed to print the detected objects)
2. Network configuration:
   - Read the network configuration from .cfg file
   - Initialize the network to zero values and dynamically allocate memory
   - Parse network configuration
   - Parse layer configuration
3. Load weights, biases, rolling mean, scales…
4. Inference/prediction
5. Get network boxes
6. Sort predictions
7. Draw detections
8. Save Image



Specific modifications according to the operation:

1. Load alphabet (images employed to print the detected objects)
2. Network configuration:
   - Read the network configuration from .cfg file
   - Initialize the network to zero values and dynamically allocate memory
   - Parse network configuration
   - Parse layer configuration
3. Load weights, biases, rolling mean, scales…
4. Inference /prediction
5. Get network boxes

A B C

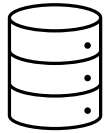


Specific modifications according to the operation:

- Avoiding the use of stlib library (fopen, malloc, qsort…)

Before ***Proposal***

Labels images:



A B C

Alphabet_cfg.h

```
#ifndef ALPHABET_CFG_H
#define ALPHABET_CFG_H

/*=============================================
*                   ALPHABET
*============================================*/

/*=============================================
*                   INCLUDES
*============================================*/
#include "darknet.h"

/*=============================================
*                   Defines
*============================================*/
#define ALP_N_SIZE          8
#define ALP_N_SAMPLES       127

extern image * alphabet_images[ALP_N_SIZE][ALP_N_SAMPLES];
extern float32_t af32_image_test[1327104];
extern image alphabet_image_test;

#endif //ALPHABET_CFG_H      You, 3 weeks ago • This versi
```

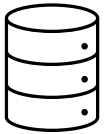
Alphabet_cfg.c

```
static float32_t af32_alphabet_0_32[198]= {1.000000f,1.000000f,1.000000f,1.000000f,1
static float32_t af32_alphabet_0_33[132]= {0.952941f,0.000000f,0.874510f,1.000000f,0
static float32_t af32_alphabet_0_34[165]= {1.000000f,1.000000f,1.000000f,1.000000f,1
static float32_t af32_alphabet_0_35[330]= {1.000000f,1.000000f,1.000000f,1.000000f,1
```

```
static image alphabet_image_0_32 = {6, 11, 3, (float32_t *)af32_alphabet_0_32};
static image alphabet_image_0_33 = {4, 11, 3, (float32_t *)af32_alphabet_0_33};
static image alphabet_image_0_34 = {5, 11, 3, (float32_t *)af32_alphabet_0_34};
static image alphabet_image_0_35 = {10, 11, 3, (float32_t *)af32_alphabet_0_35};
static image alphabet_image_0_36 = {7, 11, 3, (float32_t *)af32_alphabet_0_36};
```

```
image *alphabet_images[ALP_N_SIZE][ALP_N_SAMPLES] = {
[0] = {
&alphabet_image_0_0,  &alphabet_image_0_1,  &alphabet_image_0_2,  &alphabet_image_0_3,
&alphabet_image_0_11, &alphabet_image_0_12, &alphabet_image_0_13, &alphabet_image_0_14,
&alphabet_image_0_21, &alphabet_image_0_22, &alphabet_image_0_23, &alphabet_image_0_24,
&alphabet_image_0_31, &alphabet_image_0_32, &alphabet_image_0_33, &alphabet_image_0_34,
```

Alphabet_lib.a

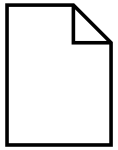


Specific modifications according to the operation:

1. Load alphabet (images employed to print the detected objects)
2. Network configuration:
   - Read the network configuration from .cfg file
   - Initialize the network to zero values and dynamically allocate memory
   - Parse network configuration
   - Parse layer configuration
3. Load weights, biases, rolling mean, scales…
4. Inference
5. Get network boxes
6. Sort predictions
7. Draw detections
8. Save Image

Previous Net and Layers Configuration

yolov7.cfg



- Network configuration

```
[net]
batch=8
subdivisions=1
width=640
height=640
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.00261
burn_in=1000

max_batches = 2000200
policy=steps
steps=1600000,1800000
scales=.1,.1
```

- Layers configuration

```
# 0
[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=swish
```

```
# 5
[route]
layers=-2
```

```
# 13
[maxpool]
size=2
stride=2
```

```
# 134
[yolo]
mask = 0,1,2
anchors = 12,16, 19,36, 40,28, 36,75, 76,55,
          72,146, 142,110, 192,243, 459,401
classes=80
num=9
jitter=.1
scale_x_y = 2.0
objectness_smooth=1
ignore_thresh = .7
truth_thresh = 1
resize=1.5
iou_thresh=0.2
iou_normalizer=0.05
cls_normalizer=0.5
obj_normalizer=1.0
iou_loss=ciou
nms_kind=diounms
beta_nms=0.6
new_coords=1
max_delta=2
```

New Network Configuration proposal

yolov7.cfg

```
[net]
batch=8
subdivisions=1
width=640
height=640
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.00261
burn_in=1000

max_batches = 2000200
policy=steps
steps=1600000,1800000
scales=.1,.1
```

Network_cfg.h

```
// 1) NETWORK ENTITIES
#define NETWORK_BATCH                       8
#define NETWORK_SUBDIVISIONS                1U
#define NETWORK_HEIGHT                      640
#define NETWORK_WIDTH                       640
#define NETWORK_CHANNELS                    3
#define NETWORK_MOMENTUM                    0.9f
#define NETWORK_DECAY                       0.0005f
#define NETWORK_ANGLE                       0.0f
#define NETWORK_EXPOSURE                    1.5f
#define NETWORK_SATURATION                  1.5f
#define NETWORK_HUE                         0.1f
#define NETWORK_LEARNING_RATE               0.00261f
#define NETWORK_BURN_IN                     1000
#define NETWORK_MAX_BATCHES                 2000200
#define NETWORK_POLICY                      STEPS
#define NETWORK_SCALES                      {0.1f,0.1f}
#define NETWORK_LEARNING_RATE_MIN           0.00001f
#define NETWORK_N                           142

#define NETWORK_SEEN                        DEFAULT_POINTER
#define NETWORK_BADLABELS_REJECT_THRESHOLD  DEFAULT_SIGNED_VALUE
#define NETWORK_DELTA_ROLLING_MAX           DEFAULT_SIGNED_VALUE
#define NETWORK_DELTA_ROLLING_AVG           DEFAULT_SIGNED_VALUE
#define NETWORK_DELTA_ROLLING_STD           DEFAULT_SIGNED_VALUE
#define NETWORK_WEIGHTS_REJECT_FREQ         DEFAULT_UNSIGNED_VALUE
#define NETWORK_EQUIDISTANT_POINT           DEFAULT_SIGNED_VALUE
#define NETWORK_BADLABELS_REJECTION_PERCENTAGE DEFAULT_SIGNED_VALUE
#define NETWORK_NUM_SIGMAS_REJECT_BADLABELS DEFAULT_SIGNED_VALUE
#define NETWORK_EMA_ALPHA                   DEFAULT_SIGNED_VALUE
#define NETWORK_CUR_ITERATION               DEFAULT_POINTER
#define NETWORK_LOSS_SCALE                  DEFAULT_SIGNED_VALUE
#define NETWORK_T                           DEFAULT_POINTER
#define NETWORK_EPOCH                       DEFAULT_SIGNED_VALUE
#define NETWORK_BENCHMARK_LAYERS            DEFAULT_SIGNED_VALUE
#define NETWORK_TOTAL_BBOX                  DEFAULT_POINTER
#define NETWORK_REWRITTEN_BBOX              DEFAULT_POINTER
#define NETWORK_LEARNING_RATE_MAX           DEFAULT_SIGNED_VALUE
#define NETWORK_BATCHES_PER_CYCLE           DEFAULT_UNSIGNED_VALUE
```

Network_cfg.c

```
// Initialization of the network using macros
network misra_network = {
    .n                              = NETWORK_N,
    .batch                          = NETWORK_BATCH,
    .seen                           = NETWORK_SEEN,
    .weights_reject_freq            = NETWORK_WEIGHTS_REJECT_FREQ,
    .equidistant_point              = NETWORK_EQUIDISTANT_POINT,
    .badlabels_rejection_percentage = NETWORK_BADLABELS_REJECTION_PERCENTAGE,
    .num_sigmas_reject_badlabels    = NETWORK_NUM_SIGMAS_REJECT_BADLABELS,
    .ema_alpha                      = NETWORK_EMA_ALPHA,
    .cur_iteration                  = NETWORK_CUR_ITERATION,
    .loss_scale                     = NETWORK_LOSS_SCALE,
    .t                              = NETWORK_T,
    .epoch                          = NETWORK_EPOCH,
    .subdivisions                   = NETWORK_SUBDIVISIONS,
    .layers                         = aptr_f32_layers,  // Initialize layers array
    .output                         = aptr_f32_output,  // Initialize output array
    .policy                         = NETWORK_POLICY,
    .benchmark_layers               = NETWORK_BENCHMARK_LAYERS,
    .total_bbox                     = NETWORK_TOTAL_BBOX,          // Initialize to
    .rewritten_bbox                 = NETWORK_REWRITTEN_BBOX,      // Initialize re
    .learning_rate                  = NETWORK_LEARNING_RATE,
    .learning_rate_min              = NETWORK_LEARNING_RATE_MIN,
    .learning_rate_max              = NETWORK_LEARNING_RATE_MAX,
    .batches_per_cycle              = NETWORK_BATCHES_PER_CYCLE,
    .batches_cycle_mult             = NETWORK_BATCHES_CYCLE_MULT,
    .momentum                       = NETWORK_MOMENTUM,
    .decay                          = NETWORK_DECAY,
    .gamma                          = NETWORK_GAMMA,
    .scale                          = NETWORK_SCALE,
    .power                          = NETWORK_POWER,
    .time_steps                     = NETWORK_TIME_STEPS,
    .step                           = NETWORK_STEP,
    .max_batches                    = NETWORK_MAX_BATCHES,
```

New Layers Configuration proposal

yolov7.cfg

```
# 0
[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=swish
```

```
# 5
[route]
layers=-2
```

```
# 13
[maxpool]
size=2
stride=2
```

```
# 134
[yolo]
mask = 0,1,2
anchors = 12,16, 19,36, 40,28, 36,75, 76,55,
          72,146, 142,110, 192,243, 459,401
classes=80
num=9
jitter=.1
scale_x_y = 2.0
objectness_smooth=1
ignore_thresh = .7
truth_thresh = 1
resize=1.5
iou_thresh=0.2
iou_normalizer=0.05
cls_normalizer=0.5
obj_normalizer=1.0
iou_loss=ciou
nms_kind=diounms
beta_nms=0.6
new_coords=1
max_delta=2
```

- Layers_cfg.h

```
/*============================================================
*                    INCLUDES
*============================================================*/
#include <darknet.h>
#include <float.h>
#include "layers_biases.h"
#include "layers_output.h"
#include "layers_rolling_mean.h"
#include "layers_scales.h"
#include "layers_weights.h"


#ifndef DEFAULT_SIGNED_VALUE
    #define DEFAULT_SIGNED_VALUE     0xAAAA5555
    #define DEFAULT_UNSIGNED_VALUE   0xFFFFFFFFU
    #define DEFAULT_FLOAT_VALUE      (float32_t) FLT_MAX
    #define DEFAULT_POINTER NULL
#endif
/*============================================================
*              PUBLIC METHODS IMPLEMENTATION
*============================================================*/
extern struct layer default_layer;
extern struct layer layer0_struct;
extern struct layer layer1_struct;
extern struct layer layer2_struct;
```

```
extern struct layer layer132_struct;
extern struct layer layer133_struct;
extern struct layer layer134_struct;
extern struct layer layer135_struct;
extern struct layer layer136_struct;
extern struct layer layer137_struct;
extern struct layer layer138_struct;
extern struct layer layer139_struct;
extern struct layer layer140_struct;
extern struct layer layer141_struct;
extern struct layer layer142_struct;

extern layer * const aptr_f32_layers[143];
```

- Layers_cfg.c

```
#include "layers_cfg.h"

/*============================================================
**                          LAYER 0
**============================================================*/
layer layer0_struct = {
    // NON-DEFAULT ENTITIES
    .type               = CONVOLUTIONAL,
    .batch_normalize    = 1,
    .n                  = 32,
    .size               = 3,
    .stride             = 1,
    .pad                = 1,
    .activation         = SWISH,
    .inputs             = DEFAULT_SIGNED_VALUE,
    .output             = DEFAULT_POINTER,
    .scales             = DEFAULT_POINTER,
    .biases             = DEFAULT_POINTER,
    .rolling_mean       = DEFAULT_POINTER,
    .weights            = DEFAULT_POINTER,

    // DEFAULT ENTITIES (comment those that are explictly defined)
    // .activation          = DEFAULT_SIGNED_VALUE,
    .lstm_activation    = DEFAULT_SIGNED_VALUE,
    .cost_type          = DEFAULT_SIGNED_VALUE,
    .forward            = DEFAULT_POINTER,
    .backward           = DEFAULT_POINTER,
    .update             = DEFAULT_POINTER,
    .share_layer        = DEFAULT_POINTER,
    .avgpool            = DEFAULT_SIGNED_VALUE,
    // .batch_normalize     = DEFAULT_SIGNED_VALUE,
    .shortcut           = DEFAULT_SIGNED_VALUE,
    .batch              = DEFAULT_SIGNED_VALUE,
    .dynamic_minibatch  = DEFAULT_SIGNED_VALUE,
    .forced             = DEFAULT_SIGNED_VALUE,
    .flipped            = DEFAULT_SIGNED_VALUE,
    // .inputs              = DEFAULT_SIGNED_VALUE,
    .outputs            = DEFAULT_SIGNED_VALUE,
    .mean_alpha         = DEFAULT_SIGNED_VALUE,
    .nweights           = DEFAULT_SIGNED_VALUE,
    .nbiases            = DEFAULT_SIGNED_VALUE,
    .extra              = DEFAULT_SIGNED_VALUE,
    .truths             = DEFAULT_SIGNED_VALUE,
```

```
# 0
[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=swish
```

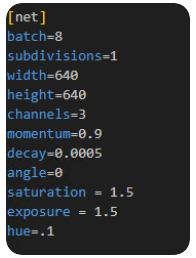
```
/*============================================================
**                          LAYER 5
**============================================================*/
layer layer5_struct = {
    // NON-DEFAULT ENTITIES
    .type       = ROUTE,
    .n          = 1,
    .n_route    = {-2, DEFAULT_SIGNED_VALUE, DEFAULT_SIGNED_VALUE,

    // DEFAULT ENTITIES (comment those that are )
    .activation             = DEFAULT_SIGNED_VALUE,
    .lstm_activation        = DEFAULT_SIGNED_VALUE,
    .cost_type              = DEFAULT_SIGNED_VALUE,
    .forward                = DEFAULT_POINTER,
    .backward               = DEFAULT_POINTER,
    .update                 = DEFAULT_POINTER,
    .share_layer            = DEFAULT_POINTER,
    .avgpool                = DEFAULT_SIGNED_VALUE,
    .batch_normalize        = DEFAULT_SIGNED_VALUE,
    .shortcut               = DEFAULT_SIGNED_VALUE,
    .batch                  = DEFAULT_SIGNED_VALUE,
    .dynamic_minibatch      = DEFAULT_SIGNED_VALUE,
    .forced                 = DEFAULT_SIGNED_VALUE,
    .flipped                = DEFAULT_SIGNED_VALUE,
    .inputs                 = DEFAULT_SIGNED_VALUE,
    .outputs                = DEFAULT_SIGNED_VALUE,
    .mean_alpha             = DEFAULT_SIGNED_VALUE,
    .nweights               = DEFAULT_SIGNED_VALUE,
    .nbiases                = DEFAULT_SIGNED_VALUE,
    .extra                  = DEFAULT_SIGNED_VALUE,
    .truths                 = DEFAULT_SIGNED_VALUE,
    .h                      = DEFAULT_SIGNED_VALUE,
    .w                      = DEFAULT_SIGNED_VALUE,
    .c                      = DEFAULT_SIGNED_VALUE,
    .out_h                  = DEFAULT_SIGNED_VALUE,
    .out_w                  = DEFAULT_SIGNED_VALUE,
    .out_c                  = DEFAULT_SIGNED_VALUE,
    // .n                       = DEFAULT_SIGNED_VALUE,
    // .n_route                 = {DEFAULT_SIGNED_VALUE, DEFAULT_S
```

```
# 5
[route]
layers=-2
```

Main operations performed in darknet-based YOLOv7

1. Load alphabet (images employed to print the detected objects)
2. Network configuration:
   - Read the network configuration from .cfg file
   - Initialize the network to zero values and dynamically allocate memory
   - Parse network configuration
   - Parse layer configuration
3. Load weights, biases, rolling mean, scales…
4. Inference
5. Get network boxes



Parse network configuration

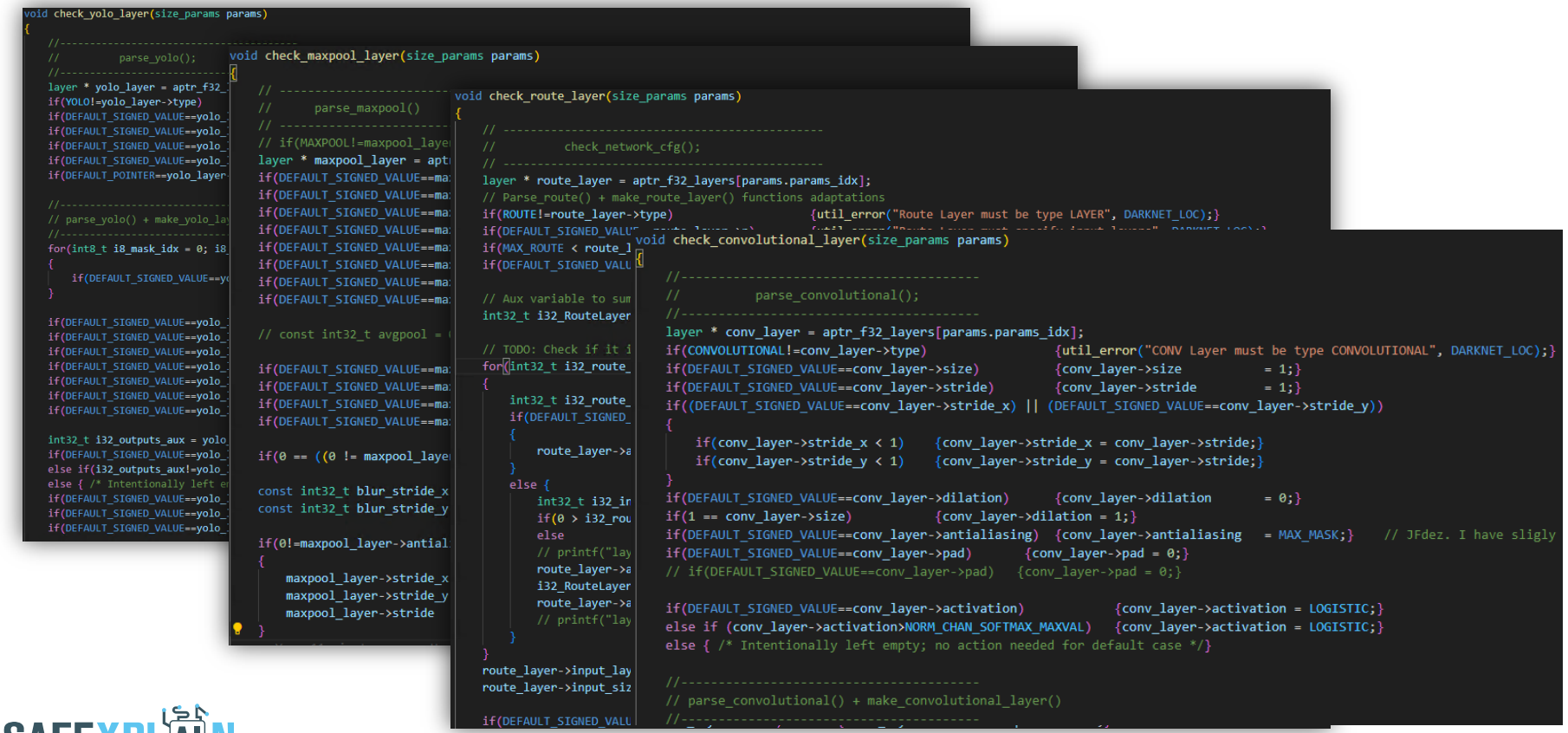
- Network configuration is done in the network_cfg.h and .c, however their values shall be parser. For that we have designed ***check_network_cfg()*** function:

```
void check_network_cfg(void)
{
    network *net = &misra_network;
    if(DEFAULT_SIGNED_VALUE == net->max_batches)       {net->max_batches   = 0;}
    if(DEFAULT_SIGNED_VALUE == net->batch)             {net->batch         = 1;}
    if(DEFAULT_SIGNED_VALUE == net->learning_rate)     {net->learning_rate = 0.001f;}
    if(DEFAULT_SIGNED_VALUE == net->learning_rate_min) {net->learning_rate_min = 0.00001f;}
    if(DEFAULT_SIGNED_VALUE == net->batches_per_cycle) {net->batches_per_cycle = net->max_batches;}
    if(DEFAULT_SIGNED_VALUE == net->batches_cycle_mult){net->batches_cycle_mult = 2;}
    if(DEFAULT_SIGNED_VALUE == net->momentum)          {net->momentum      = 0.9f;}
    if(DEFAULT_SIGNED_VALUE == net->decay)             {net->decay         = 0.0001f;}
    if(DEFAULT_SIGNED_VALUE == net->subdivisions)      {net->subdivisions  = 1;}
    if(DEFAULT_SIGNED_VALUE == net->time_steps)        {net->time_steps    = 1;}
    if(DEFAULT_SIGNED_VALUE == net->track)             {net->track         = 0;}
    if(DEFAULT_SIGNED_VALUE == net->augment_speed)     {net->augment_speed = 2;}
    if(DEFAULT_SIGNED_VALUE == net->try_fix_nan)       {net->try_fix_nan = 0;}
    if(DEFAULT_SIGNED_VALUE == net->init_sequential_subdivisions)  {net->init_sequential_subdivisions  = net->subdivisions;}
    if(DEFAULT_SIGNED_VALUE == net->sequential_subdivisions)       {net->sequential_subdivisions       = net->subdivisions;}
    if (net->sequential_subdivisions > net->subdivisions)
    {
        net->init_sequential_subdivisions   = net->subdivisions;
        net->sequential_subdivisions        = net->subdivisions;
    }
    if(DEFAULT_SIGNED_VALUE == net->weights_reject_freq)           {net->weights_reject_freq   = 0;}
    if(DEFAULT_SIGNED_VALUE == net->equidistant_point)             {net->equidistant_point     = 0;}
    if(DEFAULT_SIGNED_VALUE == net->badlabels_rejection_percentage) {net->badlabels_rejection_percentage= 0.0f;}
    if(DEFAULT_SIGNED_VALUE == net->num_sigmas_reject_badlabels)    {net->num_sigmas_reject_badlabels   = 0.0f;}
    if(DEFAULT_SIGNED_VALUE == net->ema_alpha)                      {net->ema_alpha                     = 0.0f;}
```

Main operations performed in darknet-based YOLOv7

1. Load alphabet (images employed to print the detected objects)
2. Network configuration:
   - Read the network configuration from .cfg file
   - Initialize the network to zero values and dynamically allocate memory
   - Parse network configuration
   - Parse layer configuration
3. Load weights, biases, rolling mean, scales…
4. Inference
5. Get network boxes

Parse Layer configuration



```
void check_yolo_layer(size_params params)
{
    //----------------------------------------
    //          parse_yolo();
    //----------------------------------------
    layer * yolo_layer = aptr_f32_
    if(YOLO!=yolo_layer->type)
    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_POINTER==yolo_layer

    //----------------------------------------
    // parse_yolo() + make_yolo_la
    //----------------------------------------
    for(int8_t i8_mask_idx = 0; i8
    {
        if(DEFAULT_SIGNED_VALUE==y
    }

    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_

    int32_t i32_outputs_aux = yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_
    else if(i32_outputs_aux!=yolo_
    else { /* Intentionally left e
    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_
    if(DEFAULT_SIGNED_VALUE==yolo_
```

```
void check_maxpool_layer(size_params params)
{
    // ----------------------------------------
    //      parse_maxpool()
    // ----------------------------------------
    // if(MAXPOOL!=maxpool_laye
    layer * maxpool_layer = aptr
    if(DEFAULT_SIGNED_VALUE==ma
    if(DEFAULT_SIGNED_VALUE==ma
    if(DEFAULT_SIGNED_VALUE==ma
    if(DEFAULT_SIGNED_VALUE==ma
    if(DEFAULT_SIGNED_VALUE==ma
    if(DEFAULT_SIGNED_VALUE==ma
    if(DEFAULT_SIGNED_VALUE==ma
    if(DEFAULT_SIGNED_VALUE==ma

    // const int32_t avgpool =

    if(DEFAULT_SIGNED_VALUE==ma
    if(DEFAULT_SIGNED_VALUE==ma
    if(DEFAULT_SIGNED_VALUE==ma
    if(DEFAULT_SIGNED_VALUE==ma

    if(0 == ((0 != maxpool_laye

    const int32_t blur_stride_x
    const int32_t blur_stride_y

    if(0!=maxpool_layer->antial
    {
        maxpool_layer->stride_x
        maxpool_layer->stride_y
        maxpool_layer->stride
    }
```

```
void check_route_layer(size_params params)
{
    // ----------------------------------------------
    //          check_network_cfg();
    // ----------------------------------------------
    layer * route_layer = aptr_f32_layers[params.params_idx];
    // Parse_route() + make_route_layer() functions adaptations
    if(ROUTE!=route_layer->type)                {util_error("Route Layer must be type LAYER", DARKNET_LOC);}
    if(DEFAULT_SIGNED_VALU
    if(MAX_ROUTE < route_l
    if(DEFAULT_SIGNED_VALU

    // Aux variable to sum
    int32_t i32_RouteLayer

    // TODO: Check if it i
    for(int32_t i32_route_
    {
        int32_t i32_route_
        if(DEFAULT_SIGNED_
        {
            route_layer->a
        }
        else {
            int32_t i32_in
            if(0 > i32_rou
            else
            // printf("lay
            route_layer->a
            i32_RouteLayer
            route_layer->a
            // printf("lay
        }
    }
    route_layer->input_lay
    route_layer->input_siz

    if(DEFAULT_SIGNED_VALU
```

```
void check_convolutional_layer(size_params params)
{
    //----------------------------------------
    //          parse_convolutional();
    //----------------------------------------
    layer * conv_layer = aptr_f32_layers[params.params_idx];
    if(CONVOLUTIONAL!=conv_layer->type)                {util_error("CONV Layer must be type CONVOLUTIONAL", DARKNET_LOC);}
    if(DEFAULT_SIGNED_VALUE==conv_layer->size)         {conv_layer->size          = 1;}
    if(DEFAULT_SIGNED_VALUE==conv_layer->stride)       {conv_layer->stride        = 1;}
    if((DEFAULT_SIGNED_VALUE==conv_layer->stride_x) || (DEFAULT_SIGNED_VALUE==conv_layer->stride_y))
    {
        if(conv_layer->stride_x < 1)    {conv_layer->stride_x = conv_layer->stride;}
        if(conv_layer->stride_y < 1)    {conv_layer->stride_y = conv_layer->stride;}
    }
    if(DEFAULT_SIGNED_VALUE==conv_layer->dilation)     {conv_layer->dilation      = 0;}
    if(1 == conv_layer->size)          {conv_layer->dilation = 1;}
    if(DEFAULT_SIGNED_VALUE==conv_layer->antialiasing) {conv_layer->antialiasing  = MAX_MASK;}    // JFdez. I have sligly
    if(DEFAULT_SIGNED_VALUE==conv_layer->pad)      {conv_layer->pad = 0;}
    // if(DEFAULT_SIGNED_VALUE==conv_layer->pad)   {conv_layer->pad = 0;}

    if(DEFAULT_SIGNED_VALUE==conv_layer->activation)           {conv_layer->activation = LOGISTIC;}
    else if (conv_layer->activation>NORM_CHAN_SOFTMAX_MAXVAL)  {conv_layer->activation = LOGISTIC;}
    else { /* Intentionally left empty; no action needed for default case */}

    //----------------------------------------
    // parse_convolutional() + make_convolutional_layer()
    //----------------------------------------
```

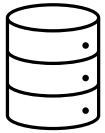
Main operations performed in darknet-based YOLOv7

1. Load alphabet (images employed to print the detected objects)
2. Network configuration:
   - Read the network configuration from .cfg file
   - Initialize the network to zero values and dynamically allocate memory
   - Parse network configuration
   - Parse layer configuration
3. Load weights, biases, rolling mean, scales…
4. Inference
5. Get network boxes

Main operations performed in darknet-based YOLOv7

Before

Yolov7.weights



Proposal

Layers_weights.h

```
#ifndef LAYERS_WEIGHTS_H
#define LAYERS_WEIGHTS_H

/*============================================
*                  INCLUDES
*============================================
#include "safeYOLO_cfg.h"
#include <stddef.h>

/*============================================
*              PUBLIC METHODS IMPLEMENTATION
*============================================
extern const float32_t af32_l0_weights[864];
extern const float32_t af32_l1_weights[18432];
extern const float32_t af32_l2_weights[36864];
extern const float32_t af32_l3_weights[73728];
extern const float32_t af32_l4_weights[8192];
```

```
extern const float32_t af32_l136_weights[1179648];
extern const float32_t af32_l137_weights[130560];
extern const float32_t af32_l140_weights[4718592];
extern const float32_t af32_l141_weights[261120];

extern const float * aptr_f32_weights[142];
```

Layers_weights.c

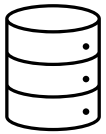
```
const float32_t af32_l0_weights[864] = {-0.016388, -0.054352, -0.020538, -0.136597, 0.449219, -0.143921,
const float32_t af32_l1_weights[18432] = {0.006386, 0.004433, 0.000838, 0.008896, -0.043732, -0.015854,
const float32_t af32_l2_weights[36864] = {0.015198, -0.007748, 0.002262, 0.011154, -0.041321, 0.004978,
const float32_t af32_l3_weights[73728] = {-0.006329, -0.029694, 0.023636, 0.022690, -0.040009, 0.004604,
const float32_t af32_l4_weights[8192] = {0.011002, 0.009552, -0.002960, -0.095642, 0.001120, 0.007896, 0
```

```
const float * aptr_f32_weights[] = {
    af32_l0_weights,  af32_l1_weights,  af32_l2_weights,  af32_l3_weights,  af32_l4_weights,
    af32_l10_weights, NULL,             af32_l12_weights, NULL,             af32_l14_weights,
```

Same for:

- Label biases
- Layer outputs
- Layers Rolling mean
- Layer scales

Weights_lib.a



SAFEXPLAIN

Main operations performed in darknet-based YOLOv7

- SafeYOLO allow to automatically generate of the following configuration documents:
  - Alphabet
  - Layer biases
  - Layer outputs
  - Layer Rolling mean
  - Layer scales
  - Layer weights

Main operations performed in darknet-based YOLOv7

1. Load alphabet (images employed to print the detected objects)
2. Network configuration:
   - Read the network configuration from .cfg file
   - Initialize the network to zero values and dynamically allocate memory
   - Parse network configuration
   - Parse layer configuration
   - Load weights, biases, rolling mean, scales…
3. Inference
4. Get network boxes

Any Question?

THANK YOU!

SAFEXPLAIN

Safe and Explainable
Critical Embedded Systems based on AI

Follow us on social media:

www.safexplain.eu

Funded by the European Union

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