This article has been accepted for publication in Computer. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MC.2024.3401402

Certifiability Analysis of Machine Learning Systems for Low-Risk Automotive Applications

Vinod Vasudevan, Amr Abdullatif, Sohag Kabir, and Felician Campean

***Abstract*—Machine learning (ML) is increasingly employed for automating complex tasks, specifically in autonomous driving. While ML applications bring us closer to fully autonomous systems, they simultaneously introduce security and safety risks specific to safety-critical systems. Existing methods of software development and systems based on ML are fundamentally different. Moreover, the existing certification methods for automotive systems cannot fully certify the safe operation of ML-based components and subsystems. This is because existing safety certification criteria were formulated before the advent of ML. Therefore, new or adapted methods are needed to certify ML-based systems. This article analyses the existing safety standard, ISO26262, for automotive applications, to determine the certifiability of ML approaches used in low-risk automotive applications. This will contribute towards addressing the task of assuring the security and safety of ML-based autonomous driving systems, particularly for low-risk automotive applications, to gain the trust of regulators, certification agencies, and stakeholders.**

***Index Terms*—Artificial Intelligence, Safety Assurance, Robustness, AI Safety, Certification, Machine Learning**

I. Introduction

Millions of new automobiles are produced annually for daily human use by the automotive industry. Because of this, the manufacturers of these automotive systems have always been increasingly concerned with ensuring their safety [1]. Furthermore, since modern automotive systems rely on increasingly complex software, sensors, actuators, and other components, their implemented functionalities have become much more complex [2].

Long-term research has been done on the use of Artificial Intelligence (AI) approaches in safety-critical systems and the utilisation of ML/ Deep Learning (DL) approaches in these systems has increased significantly in recent years. ML has proven to be an effective tool in tackling intricate problems and enhancing system performance. Nevertheless, there exist both obstacles and prospects in these domains that require careful consideration. Automotive ECUs are given additional power to make judgments and operate without driver involvement. Continuous learning from operations and dynamic reconfiguration in response to component/subsystem failures are conceivable with ML-based systems.

Although ML-based automotive driving applications have made significant advancements in recent years, they pose various challenges. Major challenges include safety and reliability, and regulatory and ethical concerns. ML models may not always perform as intended in new or unforeseen situations leading to potential risks. Moreover, ML-based systems are known to have some weaknesses that can potentially compromise safety. Some of these weaknesses include the lack of explainability in ML-based approaches, biases in the data, the integrity of data, uncertainty and confidence estimation associated with the input data, and sensitivity to adversarial inputs. These weaknesses can lead to incorrect decision-making in critical driving situations, which can pose safety hazards. Consequently, the deployment of autonomous driving technology raises various regulatory and ethical questions. Hence, the regulatory framework may need to be updated to address liability and safety standards. In the field of safety-critical applications, safety certification has become an established process over the past few decades. It serves as a means to demonstrate conformity to regulatory requirements. The growing trend of integrating ML components into safety-critical applications poses challenges to traditional certification approaches.

Analysing functional safety is a challenging task for autonomous vehicles where vehicle architecture is complex and they do not rely on human input to ensure their safety. To guarantee that such complex automotive systems achieve an acceptable level of safety, a functional safety standard, named ISO 26262 [3], has been developed. The ISO 26262 standard provides guidelines for ensuring functional safety in the development of vehicles, with part 6 specifically focusing on software implementations [4] [5]. The ISO/PAS 21448 standard, also known as SOTIF, details the iterative process of specification, development, verification and validation phases of ISO 26262. SOTIF expects that the risk of unsafe scenarios/inputs that fall under unknown and known categories is reduced to an acceptable level [5]. It also notes that a training set cannot cover all possible inputs and cannot replace a specification. ML components have unique features compared to traditional electrical and electronic components. Hence, safety analysis of ML components requires additional considerations beyond functional safety and ISO 26262 process requirements do not apply directly to AI-based software [6].

Several projects, institutes and working groups such as EUROCAE WG-114, SAE G-34, DEEL, RISE and SMILE [7] were created to address the challenges of ML reliability, trustworthiness and certification. The DEEL project conducts a comprehensive evaluation of current techniques designed to certify ML software used in safety-critical systems. However, the formal standard process for the automotive industry will take many years, whereas there is a huge demand for ML technologies in the automotive industry.

The authors are with the Faculty of Engineering and Digital Technologies, University of Bradford, Bradford, BD7 1DP, UK (e-mails: V.Vasudevan@bradford.ac.uk, A.R.A.Abdullatif@bradford.ac.uk, s.kabir2@bradford.ac.uk, and f.campean@bradford.ac.uk)

This article presents an analysis of the certifiability of low-risk (ASIL-A) ML-based systems based on the existing industry standards. To do this, we described a workflow for the development of an ML component for automotive applications, following the guidelines provided in the ISO26262 standard for software component development. The article then examines industry standards' goals and techniques to see if they can be satisfied and applied to the suggested ML component development workflow. The goal of this analysis is to make it easier to quickly implement ML technology in low-risk automotive applications using the current regulatory framework, all without the need for new legislation.

II. An Overview of ML in Automotive Applications

AI/ML has grown exponentially in numerous areas, including transportation, manufacturing, aerospace, medicine and more. Among different ML approaches, Deep Neural Network (DNN) has been widely used for object recognition and classification tasks in automotive applications. Although different ML algorithms were used for different purposes in automotive applications, for brevity, in this study, we consider DNN only. Following are some of the notable applications for DNN in the automotive industry.

- Traffic sign recognition and Lane Keep Assist.
- Vehicle trajectory analysis.
- Remote diagnosis and prognostics, failure localisation, etc.

Despite posing multiple challenges, ML approaches are already used in high-assurance systems. Depending on the use case, running ML systems in a car can affect the vehicle's security and functionality. The designers of a safety-critical ML system must deal with a wide variety of challenges, including appropriate architectural choice, trust and confidence, testing, and explainability [8].

A work from the RISE SMILE project provides a review of V&V that aims to embed ML into automotive safety-critical systems [9]. Even though these criteria were set before ML's wide usage, developers of safety-critical applications for automobiles must adhere to them. The technical report from the joint committee of EUROCAE and SAE, published in 2021, discusses the limitations of the current certification requirements in relation to ML technology and looks at alternative solutions to the problem. Another work [10] examines the difficulties of certifying systems based on ML in great depth, but without offering any answers to these problems. Several ongoing studies focus on solving certain problems in ML assurance (design, runtime, verification, and traceability). However, innovative approaches are still in their early stages, and currently, there is no comprehensive method to handle all incompatibilities in ISO 26262 [10].

III. ISO 26262 and ML

The use of ML algorithms in systems that interact with people or function in shared environments with them has led to an increase in awareness of safety concerns. Automotive safety is defined by ISO 26262 and when it comes to functional safety, the auto industry should follow this standard. The main objective of ISO 26262 is to lessen the chances of catastrophic failure of an automobile's electrical and electronic systems while still allowing them to perform their intended functionalities.

Safety and the concepts of risk, hazard, damage, and uncertainty are closely related. In a system, outcomes are produced based on its state and inputs, without any specific focus on ML. The outcome may be desirable or not, but it only becomes harmful when its consequences exceed a certain threshold set by society. Physical injury or damage to people's health, whether caused directly or indirectly, is referred to as harm. A hazard is a potential source of harm, whereas risk is the likelihood of harm multiplied by the severity of the potential harm. Unexpected or unknown events and operating conditions can have negative consequences.

Hazard Analysis and Risk Assessment (HARA) are methods that are used to identify impermissible hazards and risks that could result from failures in a vehicle's vital systems. The system's development is then directed by these safety standards. ISO 26262 uses a V-model method, and the high-level phases and their corresponding component numbers are depicted in Fig. 1 [5]. This strategy aims to ensure that all necessary software safety measures are taken during the development and testing phases. This model's objective is to guarantee that all software safety requirements included in the design are fully verified and validated [11].

Throughout its entire life cycle, the functional safety analysis system considers the ASIL levels determined for its potential threats back in the idea phase. According to one's amount of exposure, an ASIL is assigned from levels A, B, C, and D. The strictest rating is 'D,' which calls for a very low failure rate and thorough testing, while 'A' is the least demanding. For those risks that do not call for special precautions, quality management (QM) is also an option. Technical safety criteria are developed from functional safety requirements after the architecture has been defined. In contrast, functional safety criteria often outline the steps that must be taken to lessen risk to an admissible level.

The main goal of functional safety (ISO 26262) is to facilitate the industry's organised response to functional safety issues. ISO did not think about including ML in the functional safety standard until AI/ML became crucial to the automobile sector. ISO 26262 suggests traditional safety assurance methodologies, however, these are either inadequate or inappropriate for ML assurance.

Regarding the security of ML models, Salay *et al.* [12] published an analysis of ISO-26262 part-6 procedures. Based on their research, they concluded that 40% of software safety techniques cannot be applied to ML models. Therefore, ML/DNN integration with autonomous vehicles presents unique hurdles for the automotive industry. They also looked at the feasibility of applying the unit-level software approaches that are so highly recommended by ISO 26262 and discovered that 70% of them may be used to construct ML-based components directly [12]. The method proposed in [5] acknowledges and controls the ML's natural performance constraints. It opens the door to circumstances for which no training data will be available, but it gives the tools to deal with them safely.

This article has been accepted for publication in Computer. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/MC.2024.3401402

To ensure ML models are secure, several researchers have proposed various approaches. Few studies have taken into account industry norms to evaluate the security of ML components. For example, Radlak et al. [13] outlined a process for modifying the functional safety life cycle as described in ISO 26262 [3] for ML components. Certification studies have also been fueled by the current AV boom. The difficulties of ML certification are among those highlighted by Koopman and Wagner [14]. Martin et al. [15] evaluate ISO 26262's suitability for AV, albeit they pay more attention to the repercussions of the added complexity it causes than to the application of ML itself. Spanfelner et al. [16] studied ISO 26262 with a focus on the impact of its high complexity on driver-assisting devices, rather than specifically examining the use of ML.

Uncertainty plays a role in safety when a system's outcome is unknown, and there is limited or no knowledge about its probability distribution. Epistemic uncertainty specifically stems from a lack of understanding about the physical world. Safety, broadly defined, involves minimising both risk and epistemic uncertainty to prevent severe undesirable outcomes [17]. Engineering secure systems hinges on reducing the risk factor, achieved by lowering the likelihood of harmful events or minimising their impact. This necessitates a meticulous identification of potential hazards, followed by assessing their risks and implementing strategies to either eliminate or mitigate these hazards [10]. Moreover, in the majority of industries, safety case documentation is required for the efficient implementation of safety protocols. These documents serve as compelling evidence that the system meets safety objectives within a specific application and environment.

Models used in ML are grounded in statistical learning theory. In statistical ML, the fundamental concept is minimising risk. This involves having a set of observations, features (x), and labels (y), within a random example space X, Y described by a probability density function $P(X, Y)$ to find a mapping function.

$$f : X \to Y \tag{1}$$

The risk of a prediction function f is defined in statistical learning as the expected value of the loss incurred by predicting the label of an observation x as $f(x)$ instead of the y as measured by the loss function L. This is expressed as [18]

$$R(f) = \int L(x, f(x), y) dP(x, y) \tag{2}$$

In statistical learning, the main objective is to identify the function f that minimises the risk R. However in ML, the exact probability distribution $P(X, Y)$ is unknown and only a limited set of m examples $(x(i), y(i)), i = 1...m$ are available. To address this issue, we rely on the algorithm hypothesis (h) and loss function (L) to determine the expected loss on the training set, which is known as the empirical risk of h [18]:

$$Remp(h) = 1/m \sum_{i=1}^{m} L(x(i), h(x(i)), y(i)) \tag{3}$$

Defining ML safety in a formal manner poses a significant challenge. Various authors have endeavoured to delineate ML safety by referencing factors like risk, epistemic uncertainty, and the harm resulting from unforeseen outcomes. The objective of different safety methodologies in ML is to diminish undesirable results and lower the likelihood of harm by incorporating these considerations into the loss function (L), thereby enhancing the ML system's capacity for safe behaviour. Adhering to safety necessitates minimising the chance of hazardous behaviour. Compliance with functional safety standards involves demonstrating that failure probabilities fall within the range of 10^{-7} and 10^{-9} per operating hour [19]. Despite being a primary safety solution within the ML framework, the approach proposed in [18] is deemed insufficient when applied to real-world scenarios involving the coexistence of humans and machines, such as autonomous driving or industrial automation.

The challenge of distinguishing between an acceptable outcome and a potentially harmful undesired result becomes prominent when integrating ML-based control systems into safety-critical environments. It is crucial to articulate a suitable safety-focused response to identify undesired outcomes with potential harm. Effectively addressing the safety concerns of ML systems requires the meticulous implementation of processes outlined in functional safety. This approach should blend well-established safety strategies from traditional systems development with safety methodologies specifically tailored for ML systems. Bridging the gap between the risk-averse functional safety culture of conventional systems and the ML systems presents a considerable challenge that needs resolution.

IV. Analysis of ISO26262 and Proposed workflow

Here, we examine whether or not the *proposed ML development workflow* satisfies the ISO 26262 safety objectives. We zero down on ASIL A software, where just a subset of objectives is relevant, and demonstrate that these objectives may be met despite significant differences between ML-based and conventional methods of software development.

To develop the ML development workflow, we studied the current ISO26262 guidelines for the software development process as shown in Fig. 1. As seen in the figure, software architecture is made up of parts that interact with one another in a certain order. A full software system, including its architecture, might be implemented with ML. In this article, we have the following assumptions regarding the ML development process and the proposed workflow is shown in Fig. 2.

- Both ML-based software and software tailored to a particular vehicle are included in the components of a typical ML-based system. Unit-level ML components, rather than system-wide or overall architecture, are the subject of this study. Conventional software components may use ISO 26262 without any adjustments. Due to its incompatibility, full-stack ML use is currently being ruled out.
- In addition, we omitted ML systems that can train themselves while they are in operation.
- To illustrate the salient features of an ML-based system, we use DNN as an example in this study.

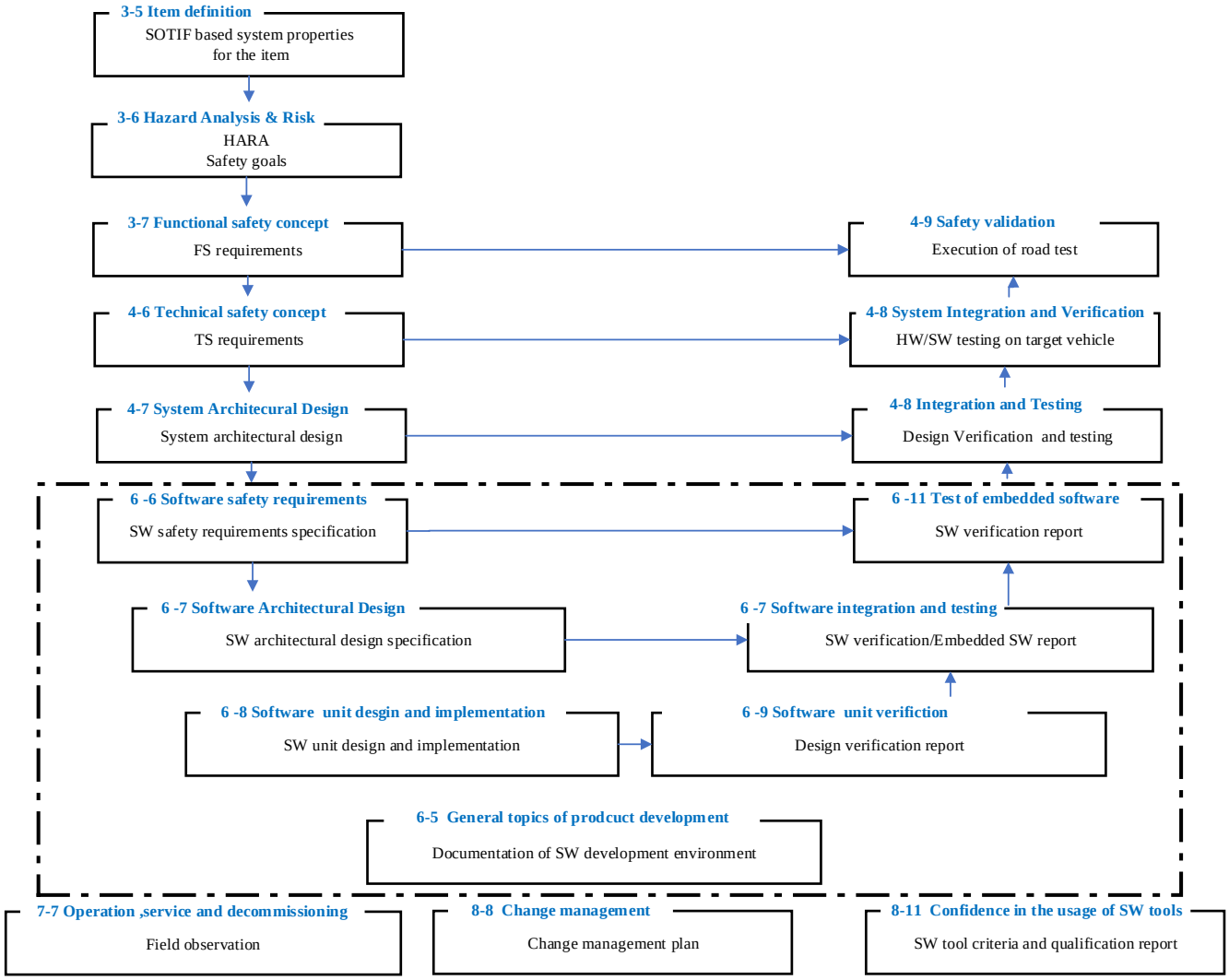

Fig. 1. V Model: ISO/PAS 21448 and ISO 26262 sub-phases [5] [13]

- To train the ML model and make it available for use, we employ the standard ML development pipeline.

A. *Proposed ML development process*

The transition from traditional programming methods to ML development necessitates an adjustment in the approach to building assurance frameworks to align with the learning process. An effective solution involves representing the ML model workflow in a manner that corresponds to the existing software development process, as depicted in Fig. 2. An example of a viable solution is the incorporation of an iterative loop during both the training and testing phases within the ML development, a crucial component integrated into the certification process.

Software requirements analysis, design, coding, and safety criteria are all part of the ISO 26262 specification, which is part of the proposed ML development process. To achieve the goals of the ISO 26262 process, several established industrial methodologies for software development flow and techniques such as the waterfall model, agile methods, model-based design or combination can be applied. All levels of requirements in the V model must be defined and developed with bi-directional traceability. However, depending on the selected development approach, the timing and connection of the development activities are changeable. In contrast to traditional software, ML models are described by a large number of parameters that are refined throughout training using a learning algorithm. In this way, it is extremely difficult to link the model's parameter settings for auto-tuning back to the underlying functions that led to those settings. One of the biggest obstacles to ML explanations is the lack of a clear paper trail. The source code of conventional software can be read by humans and traced back to the criteria it fulfils. In contrast, a learning algorithm automatically adjusts a huge number of parameters to optimise the functional behaviour of an ML model during training. ML models are usually incomprehensible to humans. It is considered that mapping and tracing the values of these automatically modified parameters in the ML model is almost impossible. Therefore,

This article has been accepted for publication in Computer. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MC.2024.3401402

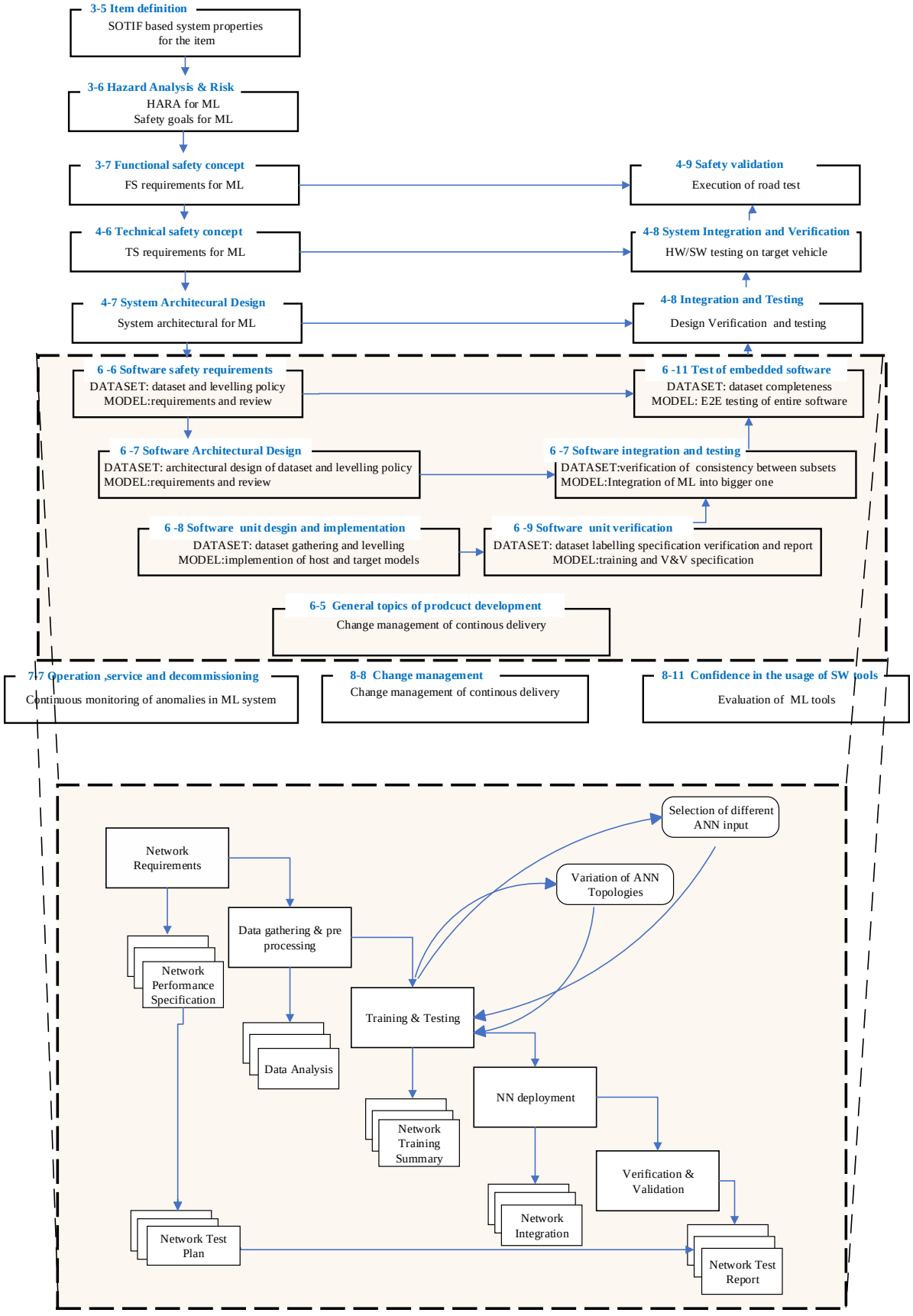

Fig. 2. ISO 26262 sub-phases and proposed aspects of ML that need to be addressed during development of ML-based components

This article has been accepted for publication in Computer. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/MC.2024.3401402

TABLE I
ISO26262 RECOMMENDATIONS FOR VERIFICATION AND TESTING OF SOFTWARE ARCHITECTURE WITH DIFFERENT ASIL LEVELS [3]

	Verification	**QM**	**ASIL A**		**ASIL B**		**ASIL C**		**ASIL D**	
	Methods		Highly Recommended	Recommended	Highly Recommended	Recommended	Highly Recommended	Recommended	Highly Recommended	Recommended
1	Walk-Through		✓			✓				
2	Inspection			✓	✓		✓		✓	
3	Semiformal verification			✓		✓	✓		✓	
4	Formal Verification							✓		✓
5	Control flow analysis			✓		✓	✓		✓	
6	Data flow analysis			✓		✓	✓		✓	
7	Static Code analysis			✓	✓		✓		✓	
8	Semantic code analysis			✓		✓		✓		✓
	Testing	**QM**	**ASIL A**		**ASIL B**		**ASIL C**		**ASIL D**	
	Methods		Highly Recommended	Recommended	Highly Recommended	Recommended	Highly Recommended	Recommended	Highly Recommended	Recommended
1	Requirements based test		✓		✓		✓		✓	
2	Interface test		✓		✓		✓		✓	
3	Fault injection test			✓		✓		✓	✓	
4	Resource usage test			✓		✓		✓	✓	
5	Back to back comparison test			✓		✓	✓		✓	

the goals of traceability for an ML model are unachievable. One potential solution involves incorporating an iterative loop into the training and testing phases of ML software development, which is necessary for the certification process. This solution is applicable with the provided dataset, where model performance is assessed during the design phase and specific performance assurances are made for the operational phase (see Fig. 2). Since ASIL A is the lowest safety level that we are considering, traceability has no effect on ASIL A software.

In addition to outlining the development process, ISO specifies supporting processes such as Part 7 (covering production, operation, service, and decommissioning) and Part 8 (supporting processes). Furthermore, it is recommended to incorporate other parts of ISO 26262 when integrating ML-based components. Continuous monitoring of product post-production (Part 7) is advised, particularly for detecting outdated ML models. The objective of this process remains technology-independent and can be applied to ML-based systems at any assurance level. Specifically, the field monitoring process should be carefully planned to determine the what, when, by whom, and how often data should be processed and updated. Field monitoring and change management should be closely integrated.

ML-based systems are expected to require continuous updates throughout their product lifespan. Thus, the planning of change management should encompass all potential trigger events necessitating modifications. These events may include scheduled continuous updates, adjustments prompted by identified anomalies, or adaptations due to evolving demands. This analysis and planning phase should occur early in the development process. Establishing robust change management planning is crucial for automating tasks by integrating the backend tool chain.

V. CERTIFIABILITY ANALYSIS

This section analyses the certifiability of ASIL A-level ML approaches in the context of the ISO26262 standard. Table I shows the recommendations provided by ISO 26262 for verification and testing requirements of software architectures at different ASIL levels. As can be seen, different verification and testing methods were either *recommended* or *highly recommended* for different ASIL levels. Due to the limited scope, our focus will be on the recommendations provided for ASIL A-level software architectures. For ASIL A-level software architectures, six verification approaches were recommended, one was highly recommended, and formal verification was not needed, whereas two testing approaches were highly recommended and three approaches were recommended. ISO26262 also provides numerical recommendations for error-handling mechanisms such as 0 (N/A), 1 (recommended), and 2 (highly recommended) for different ASIL levels. The following list shows these recommendations for different error-handling methods [3].

1) Static recovery mechanism: A (1), B(1), C(1), D(1)
2) Graceful degradation: A (1), B(1), C(2), D(2)
3) Independent parallel redundancy: A (0), B(0), C(1), D(2)
4) Correcting codes for data: A (1), B(1), C(1), D(1)

As the ML model is positioned to solely address low-level software requirements within the suggested workflow, the matter of traceability does not impact the ASIL A software. This is due to the fact that the objectives related to low-level requirements are not applicable to ASIL A software. Two significant challenges are the absence of specification and non-interpretability. Non-interpretability poses a barrier to safety assurance by hindering the application of manual white box verification methods like inspection and walk-through.

TABLE II
OUTCOMES OF THE CERTIFIABILITY ANALYSIS

ASIL level					Category			
ASIL A	**ASIL B**	**ASIL C**	**ASIL D**	**Technique**	**OK**	**Adapt**	**N/A**	**Analysis**
				Unit design Notations				
2	2	2	2	Natural language	1			Can use natural language to describe hyper parameters, training set strategy etc.
2	2	1	1	Informal notations	1			Can use informal notations to describe hyper parameters, training set strategy etc.
2	2	2	2	one entry exist point in subprograms	1			Applicable at the ML component interface level
2	2	2	2	Initialization of variables			1	Only meaningful for programming
				Unit design and implementation verification				
2	1	0	0	Walk- through		1		Adaptation: walkthrough structure of transparent ML models
				Unit Testing				
2	2	2	2	Requirements-based test	1			Black-box technique
2	2	2	2	Interface test	1			Black-box technique
				Unit Deriving Test Cases				
2	2	2	2	Analysis of requirements	1			Black-box technique
2	2	1	1	Statement coverage			1	Only meaningful for programming

Moreover, it extends its relevance to other activities such as formal verification or static analysis, where comprehension of the implementation is crucial for interpreting the results.

Part 6 of ISO 26262 outlines 75 software development techniques applied across different phases of the V model. Out of these, 34 techniques are specifically relevant at the unit level, while the remaining ones are intended for the architectural level [12]. The assumption is that ML was exclusively utilised at the unit level, focusing the analysis on techniques applicable to this level. Nevertheless, the challenges posed by the absence of specification and non-interpretability directly affect the capability to perform verification and testing. The behaviour of ML is dependent on certain parameters that are automatically updated during model training. As a result, traceability objectives are not feasible for an ML model. However, traceability has no impact on applications with low criticality.

The following methods can be achieved in traditional software. Software testing may largely be performed in a hosted or simulated environment. Such testing cannot guarantee that software requirements have been met. ISO 26262 established additional test environments for performing tests to increase the confidence that the software is operating correctly. The validation objectives according to ML technology relate mainly to the verification and coverage assessment of the ML model.

ISO26262 requires assessments of 1) Requirements coverage by tests and 2)Interface tests. At the preliminary stage, we evaluate how effectively our tests mimic the expected behaviour described in the program specifications. To do this, it is necessary to conduct a review of tests and related requirements. As we have seen, however, because it is not human-comprehensible, ML models are not usually linked to their respective requirements or tests (explainability challenge). Performing additional activities, such as detecting unexpected functionality in the source code, is necessary to achieve a greater level of trust in the correctness and completeness of requirements-based tests. When validating ML models, the verification and coverage assessment of the ML model are the main objectives related to the verification of requirement-based tests. In order to accomplish these objectives, extra activities are required to detect any unintended functionality in the source code.

ISO mandates specific techniques for various stages of software development and our analysis shows the majority of them are applicable to ML components and others could be easily adapted for low-criticality applications(ASIL-A). Table II shows the results of assessment techniques which are highly recommended for ASIL A from the 34 techniques which apply to the unit level. Each technique can be classified into three categories based on the level of applicability of ML.

Category OK is directly applicable without modification. Adapt means if the technique is modified in some manner and can be utilised in the ML component. N/A indicates that the technique is applicable to code-oriented programming and not applicable to an ML component. Analysis performed in [12] considered the ML at the system level and examined the applicability of all 34 available techniques. On the other hand, we considered ML at the component/unit level and examined only the relevant techniques applicable at a particular unit level. Although the scope and coverage of the analysis performed in [12] and our work were different, the results of our analysis are aligned with the results produced by [12].

Our analysis and the analysis performed in [12] both identified that 60% of the existing techniques can be used as they are in the current form without any special modifications for ML approaches for low-criticality applications and 10% of existing approaches would need to be adapted before they can be used.

VI. CONCLUSION AND FUTURE WORK

This paper examines the prevailing industrial standards in the development of automotive software for low-criticality ML systems. The analysis reveals inconsistencies between these standards and certain features of ML technology that may potentially impede the fulfilment of standard objectives, such as coverage, traceability, and model verification. To address these challenges, the paper proposes assumptions for an ML development workflow, specifically focusing on non-adaptive,

supervised learning systems. The planned future work involves expanding the analysis to encompass higher criticality levels and delving deeper into the challenges associated with explainability and coverage.

REFERENCES

[1] M. Gharib, P. Lollini, A. Ceccarelli, and A. Bondavalli, "Engineering functional safety requirements for automotive systems: A cyber-physical-social approach," in *2019 IEEE 19th International Symposium on High Assurance Systems Engineering (HASE)*. IEEE, 2019, pp. 74–81.

[2] S. Wagner, B. Schätz, S. Puchner, and P. Kock, "A case study on safety cases in the automotive domain: Modules, patterns, and models," in *2010 IEEE 21st International Symposium on Software Reliability Engineering*. IEEE, 2010, pp. 269–278.

[3] I. ISO, "26262-1: 2018," *Road vehicles—Functional safety—Part*, vol. 1, 2018.

[4] R. Salay and K. Czarnecki, "Using machine learning safely in automotive software: An assessment and adaption of software process requirements in iso 26262," *arXiv preprint arXiv:1808.01614*, 2018.

[5] S. Mohseni, M. Pitale, V. Singh, and Z. Wang, "Practical solutions for machine learning safety in autonomous vehicles," *arXiv preprint arXiv:1912.09630*, 2019.

[6] C. B. S. T. Molina, J. R. De Almeida, L. F. Vismari, R. I. R. Gonzalez, J. K. Naufal, and J. Camargo, "Assuring fully autonomous vehicles safety by design: The autonomous vehicle control (avc) module strategy," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2017, pp. 16–21.

[7] L. Alecu, H. Bonnin, T. Fel, L. Gardes, S. Gerchinovitz, L. Ponsolle, F. Mamalet, É. Jenn, V. Mussot, C. Cappi *et al.*, "Can we reconcile safety objectives with machine learning performances?" in *ERTS 2022*, 2022.

[8] K. Aslansefat, S. Kabir, A. Abdullatif, V. Vasudevan, and Y. Papadopoulos, "Toward improving confidence in autonomous vehicle software: A study on traffic sign recognition systems," *Computer*, vol. 54, no. 8, pp. 66–76, 2021.

[9] M. Borg, C. Englund, K. Wnuk, B. Duran, C. Levandowski, S. Gao, Y. Tan, H. Kaijser, H. Lönn, and J. Törnqvist, "Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry," 2018.

[10] H. Delseny, C. Gabreau, A. Gauffriau, B. Beaudouin, L. Ponsolle, L. Alecu, H. Bonnin, B. Beltran, D. Duchel, J.-B. Ginestet *et al.*, "White paper machine learning in certified systems," *arXiv preprint arXiv:2103.10529*, 2021.

[11] V. Vasudevan, A. Abdullatif, S. Kabir, and F. Campean, "A framework to handle uncertainties of machine learning models in compliance with iso 26262," in *Advances in Computational Intelligence Systems*, T. Jansen, R. Jensen, N. Mac Parthaláin, and C.-M. Lin, Eds. Cham: Springer International Publishing, 2022, pp. 508–518.

[12] R. Salay, R. Queiroz, and K. Czarnecki, "An analysis of ISO 26262: Using machine learning safely in automotive software," *arXiv preprint arXiv:1709.02435*, pp. 1–6, 2017.

[13] K. Radlak, M. Szczepankiewicz, T. Jones, and P. Serwa, "Organization of machine learning based product development as per ISO 26262 and ISO/PAS 21448," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2020, pp. 110–119.

[14] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 1, pp. 15–24, 2016.

[15] H. Martin, K. Tschabuschnig, O. Bridal, and D. Watzenig, "Functional safety of automated driving systems: Does iso 26262 meet the challenges?" in *Automated Driving*. Springer, 2017, pp. 387–416.

[16] B. Spanfelner, D. Richter, S. Ebel, U. Wilhelm, W. Branz, and C. Patz, "Challenges in applying the iso 26262 for driver assistance systems," *Tagung Fahrerassistenz, München*, vol. 15, no. 16, p. 2012, 2012.

[17] N. Möller, "The concepts of risk and safety," *Handbook of risk theory: epistemology, decision theory, ethics, and social implications of risk*, vol. 1, pp. 55–85, 2012.

[18] A. Pereira and C. Thomas, "Challenges of machine learning applied to safety-critical cyber-physical systems," *Machine Learning and Knowledge Extraction*, vol. 2, no. 4, pp. 579–602, 2020.

[19] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An open approach to autonomous vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, 2015.