

AI/ML for Safety-Critical Software

The Case of the Space Domain

Alberto Petrucci^{1b}, Gran Sasso Science Institute and Thales Alenia Space

Francesco Basciani^{1b} and Patrizio Pelliccione^{1b}, Gran Sasso Science Institute

// We analyze the current way of producing safety-critical software and the reference safety and assurance standards in the domain of spacecraft, namely, ECSS-Q-ST-80C and ECSS-E-ST-40. Then, we explore the readiness of correct practices to ensure that machine learning- or AI-enabled systems are safe and reliable. //



©SHUTTERSTOCK.COM/SDECORET

IN THE EVER-EVOLVING landscape of the space industry, where precision and reliability are paramount, software qualification and validation take center stage, ensuring the success of space missions. The intricate dance of spacecraft systems, navigation, and communication heavily relies on the robustness of the software that orchestrates these complex operations. As technology advances, integrating AI and machine learning (ML) into critical software introduces a new dimension to the qualification and validation processes, amplifying both the potential benefits and challenges. Right now, there isn't a widely agreed-upon method, set of methods, or tools everyone uses to develop AI/ML-based safety-critical systems. AI and ML enable spacecraft to autonomously adapt to dynamic environments, process vast datasets in real time, and make informed decisions without constant human intervention. However, integrating AI and ML modules into critical systems necessitates a meticulous approach to qualification and validation, as these technologies' inherent complexity and unpredictability introduce unique challenges. Critical AI/ML-enabled software demands an even more rigorous validation approach. Traditional methods may not fully capture the intricate decision-making processes of ML models or the adaptability of AI-enabled systems. Certifying the safety and reliability of AI/ML-enabled systems requires novel techniques beyond conventional testing. Ensuring the dependability of AI/ML-enabled critical software involves validating the individual components and the system as a whole and considering the interactions and feedback loops within the AI/ML algorithms.

Digital Object Identifier 10.1109/MS.2024.3412406

Date of publication 19 June 2024; date of current version 11 April 2025.

Standards and guidelines for AI/ML qualification in critical space software, such as ECSS-Q-ST-80C¹ and ECSS-E-ST-40,² provide frameworks for addressing the specific challenges posed by these technologies. The validation of AI/ML components involves testing the software under normal operating conditions and subjecting it to diverse scenarios, including edge cases and potential failure modes, to assess its robustness and resilience. Thanks to our strong collaboration with space domain companies, this article explores software qualification and validation in the space industry. Specifically, we explore the pivotal but still challenging role of AI and ML in critical software. It is important to highlight that our study does not cover large language models (LLMs) but, instead, focuses on classical AI/ML approaches. As we delve into this

analysis, we unravel the methodologies, standards, and challenges associated with ensuring the reliability of software systems that navigate the vast expanse of space, augmented by the power of AI and ML. Additionally, we consider AI trustworthiness, which encompasses various dimensions like engineering, ethics, and legal aspects. However, our focus in this context is on the safety engineering component.

Software Qualification and Validation in the Space Industry

ECSS-Q-ST-80C¹ outlines specific software product assurance requirements for space projects. These requirements cover quality management, framework, life cycle activities, process definition, and product quality characteristics.

The standard underscores the customer-supplier relationship in software

development, with ECSS-M-ST-10,³ providing insights into its organizational dynamics. The customer procures hardware and software components in space projects, typically in a hierarchical relationship with multiple contributing suppliers. The customer defines space system requirements, specifying performance, safety standards, and other critical characteristics. ECSS-Q-ST-80C¹ obligations are binding on the supplier, who must demonstrate compliance and provide evidence.

ECSS-Q-ST-80C aligns with ECSS-E-ST-40,² integrating product assurance into space system software engineering processes. Together, these standards comprehensively detail space software development processes. The European Space Agency's (ESA's) ISVV Guidebook⁴ further describes the scope and phases of this external activity, completing the overall process description.

Putting the Standards in Practice: An Industrial Perspective

Figure 1 represents a flowchart summarizing the critical software qualification process from an engineering and quality point of view.

All starts from the system requirements definition. This high-level description enables the extraction of key functionalities and initiates a failure modes and effects analysis (FMEA).⁵ The FMEA helps identify the criticality level of each functionality, allowing the isolation of a critical subset that requires protection. It is worth highlighting that functionalities can be implemented through software, hardware, or a combination of both. Accordingly, the chosen protection method should align with the nature of the functionality. Generally, a hardware-based protection solution is viewed as more reliable than

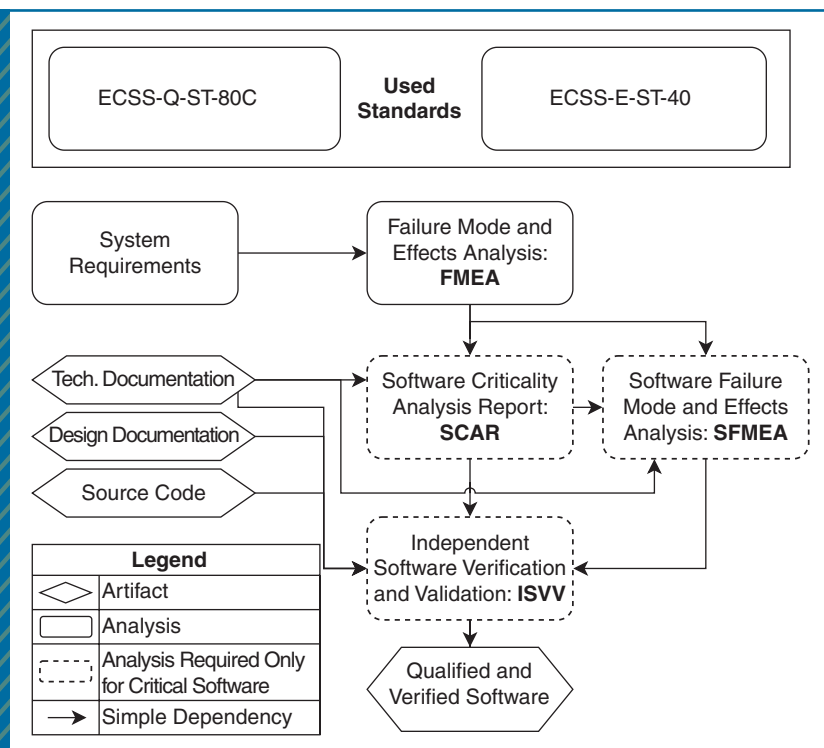


FIGURE 1. Critical (CAT-A and CAT-B) software qualification process.

a software-based one. Moving forward, utilizing the FMEA and documentation of units implementing functionality in software, the analysis progresses to generate the Software Criticality Assignment Report (SCAR).¹ This report zooms in on the criticality assessment of each primary software-implemented functionality. To delve deeper into the criticality assignment, the list of main functionalities undergoes an expansion through the software FMEA (SFMEA).⁵ Like the FMEA, the SFMEA zeros in on exploded software functions, drawing insights from the technical documentation of units implementing software functionalities.

The final step in this journey is independent software verification and validation (ISVV).⁴ This engineering practice aims to enhance software product quality, reduce costs, and mitigate development risks. An independent organization conducts verification and validation tasks, emphasizing process efficiency and complementing the supplier's efforts. Upon the completion of ISVV, the software product attains qualification.

This process follows an iterative approach. If any issues or missing information arise during any step, there is flexibility to revisit the inputs and rectify problems, ensuring a thorough and refined analysis.

In real-world applications, flight software is categorized into three criticality levels denoted by the letters A, B, and C,¹ where A indicates the highest criticality and C the lowest.

- **Criticality level C (CAT-C):** Corrupted navigation messages and signal feared events due to common mode failures without loss of satellites. Software of Category C requires procedures for 1) system documentation,

2) software development, and 3) software testing.

- **Criticality level B (CAT-B):** Loss of satellites without catastrophic consequences, temporary injuries, major property damage, and major environmental effects. In addition to the procedures required by CAT-C, CAT-B requires 1) technical and design documentation and 2) ISVV level 1

These processes and activities are consolidated and mature. However, they are challenged when applying them to AI/ML-enabled software. The development process for AI/ML solutions drastically changes from traditional software development. Therefore, when applying the processes and activities described in this section to AI/ML-enabled software, questions arise about defining design documenta-

Certifying the safety and reliability of AI/ML-enabled systems requires novel techniques beyond conventional testing.

(activity executed by an external supplier). Among other activities, ISVV level 1 requires (a) source code analysis (readability, maintainability, and conformance to standards) for inspection, (b) functionality/code traceability, and (c) unit test development, execution, and report.

- **Criticality level A (CAT-A):** Unrecoverable failures at satellite level, potential for life-threatening injuries, loss of launch site facilities, and severe environmental effects. The software of Category A is similar to Category B but requires ISVV level 2 instead of ISVV level 1. ISVV level 2 is an external activity and, in addition to ISVV level 1, it requires a detailed view (a deep review regarding code readability, maintainability, and conformance to standards it is executed) of the source code analysis and a binary code analysis (object code coverage).

tion, the role of ISVV, and the transformation of source/binary code analysis and testing procedures. The following section sheds some light on the main aspects that must be considered for qualifying AI/ML-based software.

What is Needed for Qualifying AI/ML-Based Software?

In this section, we discuss and try to answer the following questions:

- **RQ1:** What are the main differences between traditional software development and AI/ML development?
- **RQ2:** What does it mean to qualify an AI/ML solution when used on a CAT-C functionality?
- **RQ3:** What does it mean to qualify an AI/ML solution when used on a CAT-B functionality?
- **RQ4:** What does it mean to qualify an AI/ML solution when used on a CAT-A functionality?

In the remainder of this section, we first identify the main differences between traditional and AI/ML software development (RQ1), and then we discuss to what extent and how the current practices can be adjusted in order to accommodate AI/ML.

Changes Required in Technical Documentation and Design Documentation

Referring to the process described in Figure 2, the first two steps are relevant also for the AI/ML case, i.e., 1) definition of technical requirements for the intended functionality and 2) conceptual design

for the forthcoming implementation or model. In AI/ML, the design phase assumes a distinctive nature. The AI/ML design has a dual nature involving conventional textual elements (functionality and requirements definition) and the specialized modeling techniques inherent to artificial intelligence and machine learning.

Changes Required in the Software Development to Accommodate AI/ML

Figure 2(a) summarizes the main differences between the traditional software development and the AI/ML one. Traditional software development

involves coding to bring a function to life, often using model-based techniques like Matlab's Simulink for efficiency. This approach generates C/C++ code from a model, facilitating development.

In AI/ML development, the focus shifts from implementing functionality to training the model for a specific task. Consequently, the outcome is not source code anymore, but, instead, a set of parameters (hyperparameters) describing the trained model. After training, the model can execute the desired function with statistical accuracy, diverging from the deterministic nature of traditional source code.

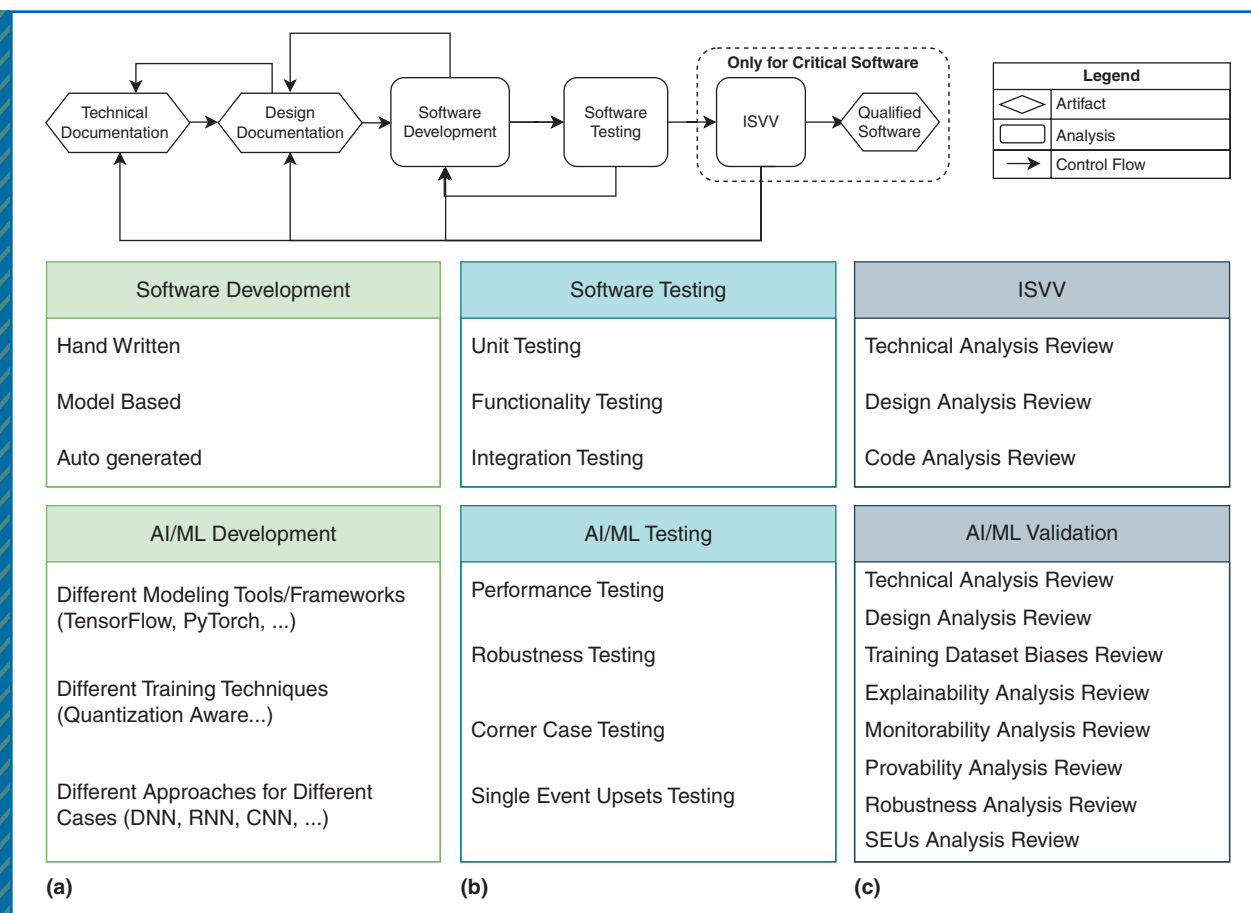


FIGURE 2. Flow chart: Current software qualification process (CAT-A, CAT-B, and CAT-C). (a) Software development versus AI/ML development. (b) Software testing versus AI/ML testing. (c) ISVV versus proposed AI/ML qualification process.

As a second point, we want to underline that with AI/ML the training dataset becomes of key importance since its content might directly affect accuracy. It becomes also important to consider how the model has been trained, e.g., if quantization-aware techniques are used or if it has been pruned.

As a final point of consideration, let us compare the efficiency of source code and AI/ML models. After choosing a CPU architecture, it is possible to measure the time it takes to run a source code using mega instructions per second. This provides a clear gauge of the time needed for the functionality. On the other hand, running an AI/ML model on a CPU can be quite challenging, especially for larger models with many inputs and outputs; they may require a hardware accelerator. The noteworthy aspect is that Xilinx has introduced various solutions specifically designed for space applications. These solutions implement hardware accelerators like neural engines, offering enhanced AI and ML model performance in challenging environments.

Changes Required in the Testing Phase to Accommodate AI/ML

Once the software has been developed, it must be tested to prove its functionality has been correctly implemented. Testing techniques are generally grouped into the white-box and black-box testing categories.

In white-box testing, there is a comprehensive understanding of how the functionality has been implemented. Testers delve into the internal workings of the software, examining the code, logic, and overall structure. This method allows for a thorough evaluation of individual components and pathways within the software. White-box testing is particularly beneficial for uncovering intricacies related to

code coverage, logical flow, and potential vulnerabilities. Black-box testing assumes the tester does not know how the functionality is implemented. Black-box testing is valuable for assessing the software from a user's perspective, emphasizing functionality and user experience without being influenced by the internal code structure.

Overall, both techniques aim to provide confidence that the functionality has been implemented correctly. Testing all the possible inputs is impractical or even impossible.

In developing AI/ML-enabled systems, the testing approach is dataset-based. So, when talking about critical functions, it is unavoidable to consider a robust and complete testing dataset that can cover most of the behaviors. Moreover, in the context of utilizing the model for space applications, executing it on hardware protected against single-event upsets (SEUs) is imperative. These are transient events caused by radiation in space that can disrupt the normal operation of electronic components. Ensuring the robustness of the model against SEUs can be addressed even during the training phase. This involves incorporating strategies and optimization techniques that make the model more resilient to the potential impact of SEUs. By considering this issue proactively, it is possible to enhance the model's reliability and performance in the challenging space environment. Figure 2(b) visually summarizes our discussed concepts.

Changes Required to ISVV to accommodate AI/ML

For safety-critical applications, a dedicated block in the qualification process involves the ISVV process, comprising three main steps:

- The technical analysis review (TAR) ensures software

requirements' correctness, completeness, and consistency with system partitioning. It also validates safety requirements using rigorous methods, aiming to identify key test cases for independent verification and validation processes.

- The design analysis review (DAR) evaluates software architectural and detailed design, emphasizing reliability, safety, error handling, interfaces, synchronization, and budget within a systematic review process. It covers two key products and phases, including software architectural and detailed design, and extends to examining the software user manual.
- The code analysis review (CAR) assesses source code, ensuring reliability, safety, error handling, proper initialization and termination, interfaces, synchronization, and budget adherence. It detects programming errors and verifies auto-coding from models, ensuring seamless integration within the software product.

In practice, TAR and DAR act as vital checkpoints, meticulously reviewing technical and design documentation for coherence, completeness, and traceability between requirements. Conversely, CAR closely examines software implementation, including dynamic processes like static code analysis. For the highest criticality levels, it rigorously examines secure code practices. The CAR ensures a direct mapping between design requirements and the implemented source code, fostering transparency and accountability in the software development life cycle.

How can these processes be translated into something applicable to

AI/ML models? We presented our proposal in Figure 2(c). TAR and DAR are present in both ISVV and AI/ML validation (for which remains valid the *auditability* properties because it is still an external activity). TAR and DAR for AI/ML-enabled systems relates to the traditional software validation. However, DAR especially introduces some challenges since it would require providing evidence that the design of the AI/ML model is consistent to the requirements of the solution to be implemented. How to realize this step in practice is (still) not completely clear.

The most significant difference emerges in the final stage, the CAR activity. In particular, some points lose their significance when applied in AI/ML-based solutions, such as source code analysis and binary code coverage. Considering only the coverage, pruning the AI/ML model to remove unused branches could be a good solution. It is not easy to replace the CAR activity for AI/ML-based systems, and there is still the need for research and study to reach the needed level of confidence and empirical evidence. To replace the code analysis activity, we need more research and experimentation to address the following challenges⁶:

- The *analysis of the training dataset*⁷ involves evaluating its completeness and alignment with the desired model functionality, ensuring it captures essential characteristics for training. This analysis is crucial for identifying and addressing biases in the dataset.
- *Explainability analysis*⁸ interprets machine learning models to understand and articulate decision-making processes. Essential for transparency, it builds

trust, ensures accountability, and meets regulatory requirements in applications where understanding model decisions is crucial.

- *Monitorability analysis*⁶ evaluates the system's ability to provide information, enabling differentiation between correct and incorrect behaviors. It measures how well the system offers insights for distinguishing between acceptable and faulty operations.
- *Provability analysis*⁶ assesses the system's capability to offer mathematical assurances regarding the satisfaction of certain properties. This involves formal verification processes to establish adherence to predetermined criteria.
- *Robustness analysis*⁹ evaluates the resilience and reliability of machine learning models under various conditions, ensuring stability and effectiveness in real-world scenarios. It aims to identify vulnerabilities, improve generalization, and enhance overall system robustness.
- *SEU analysis*¹⁰ focuses on assessing the impact of radiation-induced events on machine learning models. Addressing SEUs caused by radiation in space, the analysis aims to understand and mitigate potential errors, ensuring reliability and functionality in space environments prone to radiation-induced disturbances. New techniques, including bit-flip attacks during training, are employed to improve this issue.

Final Remarks and Discussion

So far, we answered RQ1 by highlighting the main differences between

traditional software development and AI/ML development and discussed to what extent and how the current practices can be adjusted to accommodate AI/ML. Based on what we discussed earlier, we provide an answer to the remaining questions.

- *RQ2—AI/ML for CAT-C functionality*: it can be used using the already defined methodology.
- *RQ3—AI/ML for CAT-B functionality*: CAT-B software requires some activities, like source code analysis, that cannot be easily substituted for AI/ML-based systems. There is the need for more research, soundness, and empirical evidence to build confidence that all the time constraints that the functionality must meet, considering its criticality.
- *RQ4—AI/ML for CAT-A functionality*: CAT-A software requires additional activities with respect to CAT-B software, like binary code analysis. This makes the use of AI/ML for CAT-A functionality even more difficult. Moreover, it becomes mandatory to test the model even on the cases for which it has not been trained (out of distribution testing) in order to understand how the model behaves on unpredictable data. Furthermore, it could be important to consider some redundancy inside the model architecture to grant more reliable results.

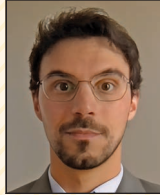
Finally, it is important to list the following open points that should be considered in addition to the presented challenges when using AI/ML in safety-critical applications:

- **OP1:** How to define an AI/ML model that can be easily extended and reused for different functionalities?
- **OP2:** How to measure AI/ML execution time?
- **OP3:** Should the training algorithm be different for CAT-B and CAT-A functions?

The initial question holds significance because it's typically feasible to expand its functionality with tested software without compromising the validity of prior tests. Is this adaptability retained in the context of AI/ML? The second query is pivotal since possessing a comprehensive dataset is the sole means to capture critical functionality thoroughly. Concerning the last question, the rationale is that there is a diversity of functionalities that can be addressed using AI/ML models, which might have different criticality and might exhibit different challenges. They could require different techniques and algorithms, as well as different training strategies and processes.

This article discussed the integration of AI and ML into safety-critical space software, emphasizing the need for rigorous qualification and validation processes. AI and ML technologies improve software capabilities by enabling autonomous adaptation, real-time processing, and informed decision-making. However, their complexity and unpredictability present unique challenges. This article suggested using robustness analysis, SEU analysis, techniques to analyze training data, monitorability analysis, probability analysis, and explainability analysis to assess AI and ML systems' vulnerabilities and enhance overall robustness. Overall,

ABOUT THE AUTHORS



ALBERTO PETRUCCI is a Ph.D. student at the Gran Sasso Science Institute, 67100, L'Aquila, Italy, and also with Thales Alenia Space. His research interests include the use of AI and machine learning in space exploration, reflecting his commitment to advancing space technology. Contact him at alberto.petrucchi@gssi.it.



FRANCESCO BASCIANI is an assistant professor in computer science at the Gran Sasso Science Institute, 67100, L'Aquila, Italy. His research interests include software engineering, model-driven engineering, and software architecture. Contact him at francesco.basciani@gssi.it



PATRIZIO PELLICCIONE is a professor of computer science at the Gran Sasso Science Institute, 67100, L'Aquila, Italy. His research interests include software engineering, architecture, and autonomous systems. Pelliccione received his Ph.D. from the University of L'Aquila. Contact him at <https://www.patriziopelliccione.com> or patrizio.pelliccione@gssi.it.

careful qualification and validation processes are crucial for the success and adoption of AI and ML in space missions. This article focused on the space domain, but we acknowledge that some findings are relevant also on other critical domains, as we can learn from previous studies¹¹; robustness, explainability, verification and validation, and certification of AI/ML are indeed shared with other domains. The certification in the space domain requires some specificities, mostly related to the ISVV, which requires, among others, source code and binary code analysis to check their readability, modifiability, and conformance to standards. 

Acknowledgment

The authors acknowledge the support of the PNRR MUR project

VITALITY (ECS00000041), Spoke 2 ASTRA - "Advanced Space Technologies and Research Alliance."

References

1. ECSS Secretariat, *Space Product Assurance. Software Product Assurance*, ECSS-E-ST-80C Rev. 1 2017.
2. ESA. "Space engineering: Simulation modelling platform." Accessed: Jun. 24, 2024. [Online]. Available: <https://ecss.nl/standard/ecss-e-st-40-07c-simulation-modelling-platform-2-march-2020/>
3. ESA. "Space project management: Project planning and implementation." Accessed: Jun. 24, 2024. [Online]. Available: <https://ecss.nl/standard/ecss-m-st-10c-rev-1-project-planning-and-implementation/>
4. ESA, *ESA Guide for Independent Software Verification and*

- Validation, ESA ISVV Guide*. Paris, France: European Space Agency, Dec. 2008.
5. ESA, *Failure Modes, Effects (and Criticality) Analysis (FMEA/FMECA)*. Noordwijk, The Netherlands: ECSS Secretariat, ESA-ESTEC, Requirements & Standards Division, Mar. 2009.
 6. J. Perez-Cerrolaza et al., "Artificial intelligence for safety-critical systems in industrial and transportation domains: A survey," *ACM Comput. Surv.*, vol. 56, no. 7, pp. 1–10, Oct. 2023, doi: [10.1145/3626314](https://doi.org/10.1145/3626314).
 7. A. Simonetta, M. C. Paoletti, and A. Venticinque, "The use of maximum completeness to estimate bias in AI-based recommendation systems," in *Proc. CEUR Workshop 8th Scholar's Yearly Symp. Technol., Eng. Math. (SYSTEM)*, 2022, pp. 76–84.
 8. R. Guidotti, A. Monreale, S. Rugieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, Aug. 2018, doi: [10.1145/3236009](https://doi.org/10.1145/3236009).
 9. R. Hamon et al., "Robustness and explainability of artificial intelligence," Publications Office Eur. Union, Luxembourg, JRC Tech. Rep., EUR 30040, vol. 207, 2020.
 10. A. Ildefonso et al., "Using machine learning to mitigate single-event upsets in RF circuits and systems," *IEEE Trans. Nucl. Sci.*, vol. 69, no. 3, pp. 381–389, Mar. 2022, doi: [10.1109/TNS.2021.3125794](https://doi.org/10.1109/TNS.2021.3125794).
 11. F. Tambon et al., "How to certify machine learning based safety-critical systems? A systematic literature review," *Automated Softw. Eng.*, vol. 29, no. 2, Nov. 2022, Art. no. 38, doi: [10.1007/s10515-022-00337-x](https://doi.org/10.1007/s10515-022-00337-x).



Call for Articles

IEEE Pervasive Computing

seeks accessible, useful papers on the latest peer-reviewed developments in pervasive, mobile, and ubiquitous computing. Topics include hardware technology, software infrastructure, real-world sensing and interaction, human-computer interaction, and systems considerations, including deployment, scalability, security, and privacy.

Author guidelines:
www.computer.org/mcl/pervasive/author.htm

Further details:
pervasive@computer.org
www.computer.org/pervasive

IEEE pervasive COMPUTING
 MOBILE AND UBIQUITOUS SYSTEMS