
# EXECUTIVE SUMMARY REPORT

## SOFTWARE PRODUCT ASSURANCE FOR AUTONOMOUS ON-BOARD SOFTWARE (PASSIONS)

| | |
|---|---|
| Prepared by: | PASSIONS Team |
| Approved by: | GMV |
| Authorized by: | Iulia Dragomir |
| Code: | PASSIONS_ESR |
| Version: | 1.0 |
| Date: | 15/11/2023 |
| Internal code: | GMV 25813/23 V1/23 |
| ESA contract number: | 4000139488/22/NL/AR/cb |

# 1. INTRODUCTION

**Autonomy** is deemed one of the most essential capabilities space systems shall include. This capability would allow for reliable operation in challenging dynamic environments, sometimes with limited bandwidth and long communication delays that restrict the possibility of direct operation from ground. For example, autonomy would be beneficial in areas such as robotics, Earth observation, spacecraft management, manned missions, among others. Future missions have stringent autonomy requirements to maximize return both in terms of operations and science.

To achieve autonomy, **Artificial Intelligence** (AI) techniques may play a crucial role. Artificial Intelligence (AI) is a ubiquitous technology in many applications domains, providing from basic functionalities such as pattern recognition to more complex ones that enable the deployment of autonomous and intelligent systems. A good definition for AI is "the capability of computer systems to perform tasks that normally require human intelligence (e.g., perception, conversation, decision-making)".

AI includes three paradigms, each aiming to provide different intelligent capabilities: symbolic AI based on rules, data-driven AI also known as **Machine Learning** (ML) based on data, and integrative AI that encompasses the capabilities of both symbolic and data-driven paradigms. ML is a novel approach in which programs are learning from data and observations of the environment, autonomously assessing the best action to take or decision to make, without an explicit definition of low-level rules or complicated sets of algorithms. There are three different approaches of ML: supervised learning in which algorithms are trained on data with labels, unsupervised learning in which algorithms are trained on data without labels and reinforcement learning in which the algorithms are trained using a reward/punishing approach for the desired/undesired behaviours. The main inconveniences of ML are its dependency on data of good quality and its opacity in the results provided (how the result has been achieved, with what uncertainty it is provided), leading to a lack of trust.

The current state-of-the-art makes use of ML for most autonomous functionalities, with great advancements in everyday (terrestrial) applications such as with self-driving cars. ML is also gaining momentum in space on-board software, with such solutions recently deployed in satellites mainly for Earth observation. Several activities are investigating the use of ML for e.g., optimal control, fault detection, isolation and recovery (FDIR), landing and obstacle detection, but also its safe deployment through (formal) verification and validation given the critical nature of these systems.

The scope of this activity is the use of **ML in on-board software**. Due to the characteristics of ML, several **challenges** need to be addressed for the safe deployment, related to development, validation, verification and quality assurance:

- **Data quantity and quality**. The performance of an ML model is directly related with the data sets used for training, validation and testing, both in terms of quantity and quality. The evolution in response to new data is amplified for ML, for which retraining can substantially alter the behaviour of the system and all its related parameters. Additionally, the data used in the development and operations process shall be validated since errors can adversely affect the quality of the model and of its performance [RD.6].
- **Model correctness**. The estimations/predictions provided by the ML system greatly impact the overall correctness, and hence assurance, of the entire system. Here correctness can be understood through the plethora of properties related to safety and dependability, reliability, resiliency, availability, maintainability, etc. One approach consists of validating and verifying (V&V) the ML model to gain guarantees of its correctness. This comes with its own challenges related to the formalization of correctness, inherent complexity of the models and V&V techniques, and its automation. Another approach consists of mitigating possible harmful results from the ML system on-the-fly through architectural elements such as redundancy schemes, validity checkers on inputs and outputs, safety cages and human-in-the-loop.
- **Model uncertainty and transparency**. The applicability of a result from an ML model is often related to its prediction uncertainty [RD.7]. In case the prediction uncertainty is low, using the result can lead to fault and/or failures which will impact on the system resilience. Uncertainty can be due to aleatory (statistical) effects, which are inherent in ML algorithms, or to epistemic uncertainty (lack of knowledge) [RD.8]. In such cases probabilistic assessments can be used to gain assurance of the model correctness, besides the uncertainty quantification through Bayesian inference of Markov models Monte Carlos simulations. Another challenge to address here is related to the typical "opaqueness" of such systems. Two "similar" input data points (from a statistical or human

  

perspective might cause the ML model to produce substantially different outputs, denoting a lack of robustness which could be exploited for security attacks, or cause serious failures which are hard to detect and/or explain [RD.9][RD.10]. This challenge can be addressed through explainability for the system behaviour [RD.11] in a manner that allows the user to understand it and the logic behind the reasoning, hence leading to acceptance validation.

- **Data and model confidentiality, integrity, privacy, and availability**. These aspects pose challenges to the assurance of ML not only in development but also in operations and the quality of the data and model needs to be assessed continuously and accordingly, besides registering and mitigating the risk prior and during operations with a risk management framework.
- **Quality model for data and models**. A quality model defines the main characteristics, metrics and possible target values which guides the acceptance and qualification of a system. In the case of ML, it is clear that two quality models are to be used: one to ensure of the data and another to ensure of the ML model/software. Ongoing work is standardizing aspects of the quality models towards a common definition of properties with possible metrics per the state-of-the-art practices [RD.11][RD.12].
- **System development assurance**. The ML lifecycle is based on an iterative paradigm of training, evaluation, optimization, validation, implementation and deployment. This is in contrast with space software development paradigm supported by the ECSS standards. A unified lifecycle that covers the main software activities is needed such that the assurance framework covers also these types of systems. Evidence needs to be gathered with respect to the data, trained model (i.e., learning), implemented model (i.e., implementation) and operations.

The "Software Product Assurance for Autonomous On-Board Software" activity aims investigating existing techniques with respect to the areas identified above and proposing a methodology that ensures the safe and reliable development and deployment of autonomous and intelligent on-board software.

  

# 2. OBJECTIVES

To achieve the goal of the activity, a set of objectives is proposed [RD.1] as follows:

1) Survey the current and planned use of on-board autonomous systems in current and upcoming European missions. This objective is twofold. On one hand, the state-of-the-art identifies the autonomous and intelligent functionalities implemented in space systems and the means to achieve them, such as traditional closed-loop algorithms or AI. On the other hand, the state-of-the-art allows to identify the impact autonomy and intelligence has on the missions, e.g., performance, and the associated needs in terms of safety, dependability and reliability.

2) Survey ongoing industrial practices and standardisation activities in the area of software product assurance for autonomous and intelligent systems, applied to other non-space safety-critical related application domains. This objective is also twofold. Firstly, the state-of-the-art reports on the techniques proposed in the literature for the verification, validation, and safety assurance of ML-based components and systems. Secondly, it reports on the standardization efforts and results for ML by different organisms, considering the safety, dependability, quality assurance and quality model aspects. Experience reports from other domains, such as automotive, are included.

3) Define requirements, methods and tools to be used for the assurance of space autonomous and intelligent on-board software, responding to the challenges presented in Section 1. In consequence, this objective is four-fold, addressing the following aspects: (i) challenges and solutions for the development, verification, validation and safety assurance of ML-based systems, (ii) safety and dependability strategies for ML-based systems, (iii) software and data quality metricated model, and (iv) requirements tailoring to software criticality levels. The solution incorporates on the above aspects the practices for ML-based systems with space software practices defined in ECSS standards and handbooks, in a unified methodology.

4) Elaborate a preliminary set of example requirements to serve as input for the revision of the ECSS standards and the related handbooks. The impact of the defined approached on the ECSS artifacts is assessed and the most relevant standards are identified, including changes to the existing requirements to encompass both traditional and ML-based software.

  

# 3. RESULTS AND LESSONS LEARNED

The activity has addressed the above objectives, proposing a methodology based on current practices for the development and deployment of on-board ML-based software, and considering it in possible revisions of ECSS standards and handbooks.

More specifically, the following have been achieved for the objectives set.

On the first objective, a non-exhaustive state-of-the-art on the use of autonomy and intelligence in on-board software space systems is performed. In the first place, it reports on the use of traditional closed-loop and ML algorithms in current and future missions executed by different space agencies, categorizing them by scope. Secondly, it identifies the safety and dependability related challenges for on-board ML and associates their impact per mission scope. The analysis is completed with an assessment of the impact autonomy/ML has on the mission performance to identify future needs. The report shows an initial acceptance of the use of ML on-board, even though the deployment of such solutions is mainly found in low-level critical systems. The adoption for high-level safety-critical systems is however still challenging, since the available techniques and tools need further maturation to support their acceptance.

On the second objective, a non-exhaustive state-of-the-art on the techniques and standards proposed for the software product assurance of ML-based systems is performed. The survey covers (i) verification and validation techniques, (ii) standards for safety and dependability and their application in other domains, (iii) standards – in work and published – and literature results for quality assurance and model and their application in other domains. The state-of-the-art allowed to identify the most relevant references for the aspect listed above that were further studied and assessed in proposing the software product assurance framework for ML-based systems. However, it is worth noting that these approaches and documents are still under evolution and consolidation.

On the third objective, a methodology based on ECSS standards is proposed covering aspects such as development lifecycle, V&V approaches, safety and dependability assurance, software and data quality models, and tailoring based on criticality level. We discuss in the following the results for each.

The ECSS-based system development lifecycle (SDLC) unifies the traditional software development lifecycle with ML development practices, including and evolving the main concepts for both: data, model, and software. Hence it proposes dedicated data and model lifecycle with milestones and deliverables, illustrated in Figure 1. The data lifecycle follows throughout the entire lifetime of a mission, from the analysis of the system-level software requirements to operations, to obtain quality data sets for training, evaluation, and testing. The model lifecycle follows the same approach for the obtention of a qualitative trained/inference model integrable and deployable on-board. Both data and model dedicated activities are performed until the end of life of the system/mission based on the needs during deployment or operations, e.g., availability of new data or optimization through partial re-training. The SDLC is considered feasible and adoptable since it is based on existing practices. It is however understood that certain aspects are challenging and need to be further addressed in the literature.

The V&V guidelines are based on the assessment of dedicated techniques for ML-based systems and their properties (functional, performance, scalability, generalization, reliability, resilience, safety, operational, etc.). The assessment considered the most promising approaches for formal verification, on one hand, and validation and testing, on the other. It included running the technique on available use cases and evaluating the results with respect to completeness of the technique, tool features such as automation, performance, and development status, and applicability to on-board software. Based on the experiments performed, we concluded that: (i) the techniques available for component-level verification have a narrow scope, dedicated to specific ML architectures and not easily extending to other ones, (ii) the tools supporting them are lacking direct usability, either needing to be run in specific configurations or to be tailored such that other similar uses cases can be injected, (iii) these tools are lacking maturity and scalability for industrial-grade use cases, with the scalability versus effort threshold not easily identifiable, and (iv) system-level verification is instead promising although the state of the art techniques are not yet integrated into standard development workflows.

For safety and dependability assurance, different techniques to gain confidence or correct the ML model predictions were assessed. From the safety perspective, the techniques surveyed are mainly related to ensuring it through architectural elements such as redundancy, FDIR and safety cages. This approach is deemed suitable for space systems, using more complex architectures for high-level safety-critical systems at the expense of effort and less complex ones for the others. This approach is to be reviewed once the V&V approach and techniques achieve the desired maturity. From the dependability perspective, the techniques surveyed aim to assess the quality of the ML model with respect to the data

  

sets used for training/validation/testing and to model prediction uncertainty, explainability, and continual learning. It must be noted that each of these topics represents open and relatively young research subjects when connected with Deep Learning models. In this sense, no out-of-the-box solutions are available, but the techniques only apply with tailoring for the system under development. Generally speaking, these approaches are considered useful and should be applied according to the scope of the ML-based system, requirements, criticality and architectural decisions.

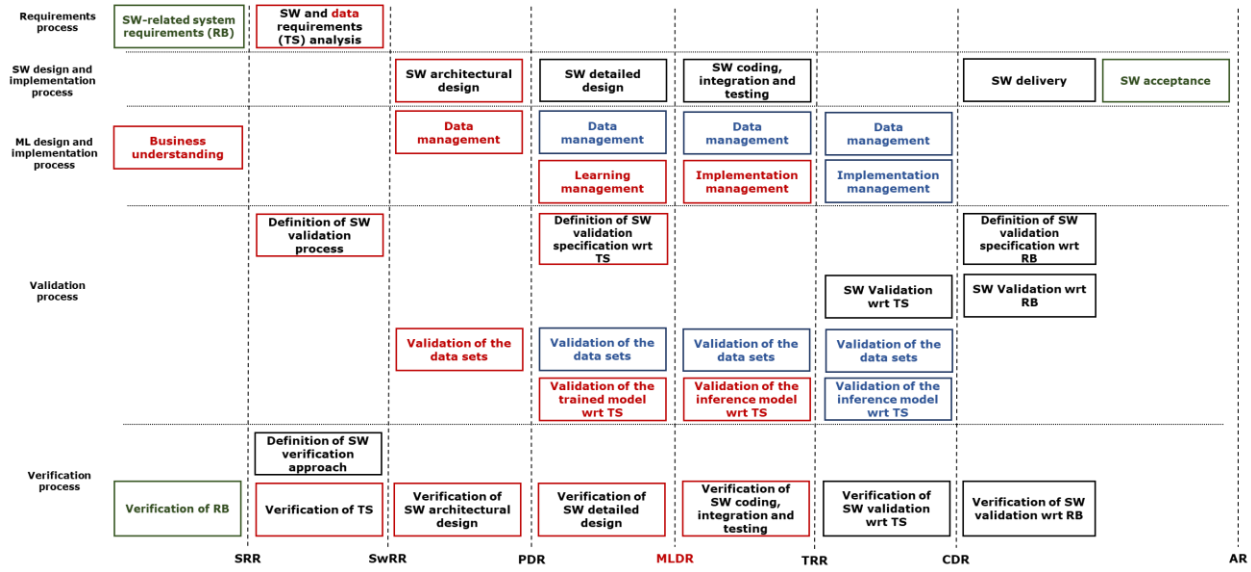


**Figure 1. Overview of the development lifecycle: ECSS activities and milestones adapted for ML. The following colour representation is used: in green the ECSS activities performed by the customer, in black the ECSS activities performed by the supplier, in red the extended or new ECSS activities for ML performed by the supplier, in blue the optional ECSS activities for ML performed by the supplier in non-nominal conditions (e.g., new available data, further optimization of the implemented model).**

An ECSS-based quality model for data and software to ensure the product assurance of the ML-based system development process is proposed. Following the new SDLC, the extended quality model covers all qualitative aspects of the new concepts – data and model. The data quality model is built upon the practices and recommendations of ISO/IEC and it completely defined with characteristics and corresponding metrics. The ML quality model is part of the software quality model. The ECSS software quality model, illustrated in Figure 2, includes characteristics specific of ML such as performance, robustness, explainability, among others. These properties have been added based on recommendations from already defined standards of ISO/IEC. The model is completed with representative metrics where applicable, both for already existing and newly introduced characteristics. These metrics are either quantitative formulas, checklists or risk-based approaches, with proposed target values based on criticality level. In building the ML-oriented quality model, some major lessons learned have been identified:

1. Context and use case must be clear. There are many application fields and potential ML-based solutions available. It is very important to be as clear as possible about the general application context. ML components should never be used just for the sake of being fancy, but always because they will add concrete value for the application context. The quality properties that are important mainly depend on this.

2. It is easy to talk about abstract generic quality properties, such as those defined by ISO/IEC 25010, on a high level. To define meaningful quality properties, we had to break them down into concrete qualities of entities and define how to operationalize these properties with measures.

3. Even though there are defined processes for ML model building and for software engineering an integrated process is missing, nor do guidelines exist on how to bring everything together with a clear focus on the quality of ML systems.

  

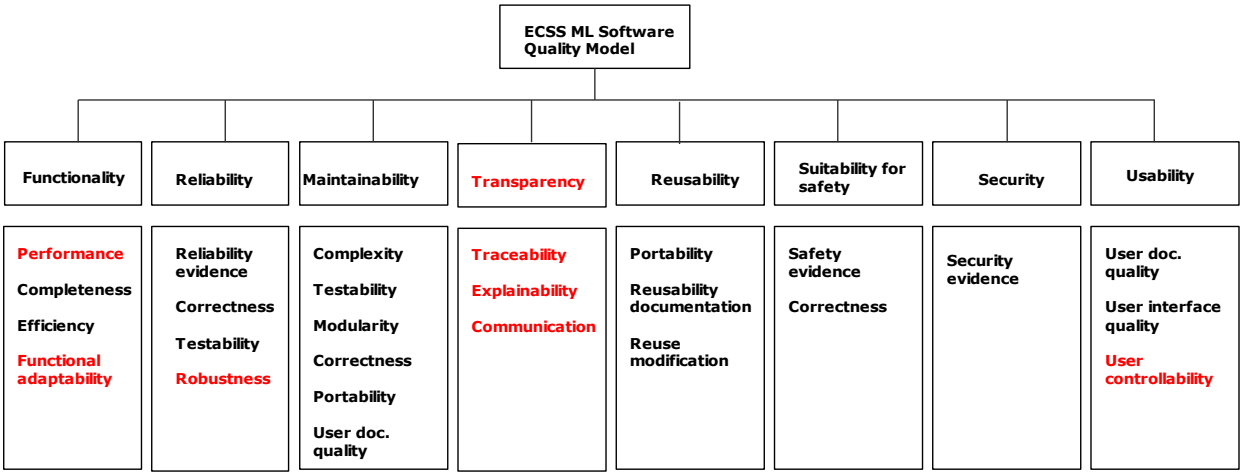


**Figure 2. Proposed ECSS ML-oriented software quality model.**

Finally, a tailoring of the guidelines based on the criticality level is performed. The criticality analysis has been evaluated from the perspective of ML-based systems, and no significant changes are identified. The same processes, as for traditional software, are applicable in this case where the concepts should have a broader understanding. The criticality level assigned comes, however, with different requirements to be ensured. The proposed solution is the employ a wide range of techniques for high-level criticality systems. For example, the use of formal verification is deemed mandatory for such systems, always augmented by test. Given the limitations explained above, the ML-based component shall be contained in a safety cage to minimize the impact of any error. Additionally, fault tolerance methods such as redundancy, can be applied to improve the system safety in contingency situations via, e.g., traditional software. Dependability techniques are also to be used to quantify the error. For low-level criticality systems, a selection would be sufficient to ensure the compliance with the requirements. For instance, the use of (formal) validation techniques (e.g., simulation, different types of testing) is deemed sufficient, given their complementation with the system-level testing.

On the fourth objective, an impact assessment of the ML-based methodology on the ECSS standards is performed. The relevant standards on the software engineering, safety and dependability, and quality assurance and metrication that need adaptation for ML-based systems have been identified. For each of these standards, guidelines on the process application are produced as request changes that could serve as input for their future revision. These changes are applicable on the current version. It is known that several standards are under revision, which implies that the change requests need to be re-evaluated once the revised versions are published. If these changes are significant, the results of this step would be only partially applicable, and a new impact assessment needs to be performed.

In general, we consider the proposed approach applicable for future space on-board software development. It is worth mentioning though that the methodology has not yet been applied on real-life use cases. The next steps include in the short-term its evaluation on different space use cases and assessment of the results obtained. In the long-term, the approach would need to be consolidated based on return of experience and new results obtained in the literature that will bring general-purpose applicability. We can also consider digitalizing it, by providing the tools and techniques to automate all the steps and producing the product assurance reports, among other useful documentation and reports.

  

# REFERENCES

[RD.1] Software Product Assurance for Autonomous On-Board Software (PASSIONS) Statement of Work, contract number 4000139488/22/NL/AR/cb 16/01/2022

[RD.2] PASSIONS Technical Note 1 "Identification of Current and Future Needs for Space On-Board Autonomous and Intelligent Systems", v1.1, 17/04/2023

[RD.3] PASSIONS Technical Note 2 "State-of-the-Art Software Product Assurance for Autonomous and Intelligent Systems in Non-Space Application Domains", v1.1, 14/04/2023

[RD.4] PASSIONS Technical Note 3 "Software Product Assurance Guidelines for Autonomous and Intelligent Systems", v1.2, 15/11/2023

[RD.5] PASSIONS Technical Note 4 "Impact Analysis on ECSS Software Product Assurance Requirements and Guidelines", v1.0, 15/11/2023

[RD.6] Gao, J., Chunli, X., Chuanqi, T. Big Data Validation and Quality Assurance – Issues, Challenges and Needs. 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE), Oxford, UK, 2016

[RD.7] Psaros, A.F., Meng, X., Zou, Z., Guo, L., Karniadakis, G.E. Uncertainty Quantification in Scientific Machine Learning: Methods, Metrics, and Comparisons, 2022, arXiv:2201.07766.

[RD.8] Hullermeier, E., Waegeman, W. Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods, 2020, arXiv:1910.09457

[RD.9] Gopinath, D., Katz, G., Pasareanu, C.S., Barrett, C. Deepsafe: A data-driven approach for assessing robustness of neural networks. In International Symposium on Automated Technology for Verification and Analysis, pages 3–19. Springer, 2018

[RD.10] Wicker, M., Huang, X., Kwiatkowska, M. Feature-guided black-box safety testing of deep neural networks. In International Conference on Tools and Algorithms for the Construction and Analysis of Systems, pages 408–426, 2018

[RD.11] Or Biran and Courtenay Cotton. Explanation and justification in machine learning: A survey. In IJCAI-17 Workshop on Explainable AI (XAI), page 8, 2017

[RD.11] ISO/IEC 5259-2 Data Quality Measures. Under preparation, 2023

[RD.12] ISO/IEC DIS 25059:2023 - Software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality model for AI systems. 2023

  

END OF DOCUMENT

  