Army Airworthiness Machine Learning Certification Supplement:

Machine Learning Development and Verification Addendum to Software Item Development

7 JAN 2025

Prepared for:
US Army Combat Capabilities Development Command
Aviation & Missile Center (DEVCOM AvMC)
Systems Readiness Directorate (SRD)
Redstone Arsenal, AL 35898-5000

Prepared by: Modern Technology Solutions, Inc. (MTSI)
360 Quality Cir NW Suite 250, Huntsville, AL 35806

Project: SRD Software Airworthiness
Phase 2 – AI Assurance (Revision 2)

Executive summary:

Army Airworthiness Machine Learning (ML) Certification Criteria (AAMLCC): Machine Learning Development and Verification Addendum to Software Item Development initially established during the AI-2363 Phase 2 builds on the foundation established by and addresses the pathologies identified in the AI-2363 Phase 1 report. Thus, raising concerns associated with ML data-driven algorithms. This document attempts to mitigate those concerns by proposing a framework, processes, objectives, activities, and output data items. The framework, processes, objectives, activities, and data items are proposed to establish the assurance and confidence in the safe application of ML data-driven algorithms in flight-critical applications. While most of the assurance practices used for traditional software items, e.g., those listed in guidance similar to RTCA DO-178C, remain applicable for ML data-driven algorithms, some assurance practices are no longer possible or their meaningfulness is being debated, e.g., structural coverage analysis. Where practices are no longer applicable, new practices that provide equivalent levels of assurance are proposed. The resultant (hybrid) assurance approach appropriately leverages the existing assurance practice, while adding new practices. The new practices, added for ML data-driven algorithms, prepend to the existing assurance practices to form the hybrid approach for ML. This document captures the status of the proposed hybrid approach as a Machine Learning Development and Verification Addendum to Software Item Development. This document serves as an acceptable means of showing compliance with Army Airworthiness requirements of ML-based software items for theArmy Airworthiness Certification Criteria (AMACC).

Epigraphs:

- “Airworthiness - The property of an air system configuration to safely attain, sustain, and complete flight in accordance with approved usage limits.” [Reference: DODD 5030.61 DoD Airworthiness Policy]
- “Airworthiness determination - The process of assessing the capability of the aircraft system and/or subsystem to meet the approved airworthiness requirements throughout the system and/or subsystem lifecycle.” [Reference: Army Regulation (AR) 70-62]
- “Airworthiness assessment - A technical evaluation of data against specific airworthiness criteria and determination of residual risk.” [Reference: DODD 5030.61 DoD Airworthiness Policy]
- "Engineering evaluation - A systems level engineering analysis to verify that the configuration and limitations of the aircraft and air systems are airworthy with respect to the airworthiness criteria defined by the airworthiness authority.” [Reference: DODD 5030.61 DoD Airworthiness Policy]
- “All models are wrong, but some are useful.” [Reference: George E. P. Box]

Table of Contents

Table of figures

Section 1 Introduction

In the context of this document Machine Learning (ML) is defined as a subfield of computer science where an algorithm is trained on a data set to produce a model that can be used to predict future values.[1,2] ML data sets are assumed to have the following characteristics:

- Data set can be used to train the Machine Learning algorithm to produce an ML model. The data set may need to be pruned or processed for it to be in the proper format.
- Data sets may be augmented, e.g., for imaging problems, where augmentations may include rotations, zooming, panning, cropping, lighting distortions, color distortions, or alpha blending of images.
- Data set is aligned with associated requirements.
- Data set is of the form and type used for direct analysis or behavior evaluation as supported by the ML model development process or the ML verification process.
- Data set may include or be composed of attributes, features, signals, and sources, and may be discrete, e.g., single sample, or temporal, e.g., simulation environment.

Due to the potential for the introduction of uncertain behavior, the following are not recommended data set approaches:

- Unstructured and unlabeled data sets.
- Adaptively collected data sets[3,4].
- Non-data-driven collected data sets, e.g., knowledge-driven.

Guidance in this document does not specifically cover reinforcement learning, unsupervised learning, or generative AI. Follow-on work is planned to address reinforcement learning, unsupervised learning and generative AI. If guidance for reinforcement learning, unsupervised learning, or generative AI is needed, coordinate with the airworthiness authority.

Within this document the ML process and ML algorithms are assumed to have the following characteristics:

- The ML algorithm parameters are populated by training techniques that use a data set(s).
- Using data sets during the ML development process, the algorithm parameters and hyperparameters are fine tuned.
- A holdout data set is used to test the parameterized ML algorithm to determine its performance on unseen data.

---

[1] Arora, S., "Mathematics of Machine Learning: An introduction", Princeton University Computer Science, Institute for Advanced Study, Plenary talk at International Congress of Mathematicians 2018, [URL: https://unsupervised.cs.princeton.edu/ICMtalk/aroraplenary.html, Accessed: 10/7/2022]

[2] Mohri, M., Rostamizadeh, A., and Talwalkar, A., Foundations of Machine Learning, Second Edition, MIT Press, Dec 25, 2018. [URL: https://cs.nyu.edu/~mohri/mlbook/, Accessed: 8/23/2022]

[3] Zhan, R., Ren, Z., Athey, S., & Zhou, Z. (2021). Policy learning with adaptively collected data. arXiv preprint arXiv:2105.02344, [URL: https://arxiv.org/abs/2105.02344].

[4] Zhang, K., "Why the sample mean can be biased on adaptively collected data (three sample illustration)", online. [URL: https://kellywzhang.github.io/assets/threesample.pdf, Accessed: 10/7/2022]

- The trained and validated parameterized ML algorithm, i.e., the Machine Learning model, is used to produce the description document used for the implementation process.

ML is inherently statistical and probabilistic, so by its nature includes uncertainty. Some ML techniques can minimize uncertainty by enforcing algorithmic limitations and restrictions, e.g., only using supervised learning techniques, while others include increased uncertainty. Due to the introduction of additional uncertainty behavior[5], the following are not recommended ML algorithm approaches:

- Reinforcement Learning, Unsupervised Learning, and generative AI.
- Adaptive learning, which is synonymous with online and continuous learning.

Possible exceptions are:
(1) the use of unsupervised Machine Learning to facilitate selection of parameters within a supervised algorithm[6], and to perform dimensionality reduction on the data set features that are then used in supervised learning, and

(2) the use of generative AI as a tool for the automated accomplishment of traditional software item processes, objectives, and activities.

Should these exceptions be used, disclosure to and coordination with the certifying authority is necessary. During the planning process, in coordination with the airworthiness authority, these exceptions will be treated on a case-by-case basis. In general, the assurance guidance presented in this document is not robust enough to endorse the use of those exceptions or other learning approaches, so the guidance presented here would need to be appropriately augmented for them to be considered.

Since the publication of traditional airworthiness guidance standards, e.g., RTCA DO-178C, advances were made in ML model development, application, verification, and tools. As the use of this technology expands into the area of safety-critical software airworthy applications, several uncertainty issues with ML need to be addressed to ensure the safe deployment of ML technology. Clarifying the approach to mitigate these issues will ease the application of ML development and verification; therefore, this document provides an acceptable means of showing compliance for application and certification approval authorities to facilitate the use of this technology.

The framework, processes, objectives, and activities in this document should be followed whenever an application uses ML data-driven algorithms. That is, guidance of this document should apply when any design or development techniques that involve the following criteria are used[7,8]:

[5] In this paper, Annex E "Frequently Asked Questions", provides addition information about why these are a cause for concern for flight critical programs, and not recommended at this time.
[6] Redmon, J., Farhadi, A. YOLO 9000: Faster, Better, Stronger, arXiv: 1612.08242, [URL: https://arxiv.org/pdf/1612.08242.pdf].
[7] Nagy, B., (2022) Level of Rigor for Artificial Intelligence Development, Naval Air Warfare Center Weapons Division (NAWCWD), China Lake, CA USA, April 2022. (NAWCWD TP 8864, publication UNCLASSIFIED.) [URL: https://apps.dtic.mil/sti/pdfs/AD1173626.pdf, Accessed: 10/31/2022]
[8] Nagy, B., Sivapragasam, G., Edwards, L. (2021) Functional Hazard Analysis and Subsystem Hazard Analysis of Artificial Intelligence/Machine Learning Functions Within a Sandbox Program, Paper presented at the Proceedings of the Eighteenth Annual Acquisition Research Symposium. [URL: https://dair.nps.edu/handle/123456789/4401, Accessed: 10/31/2022]

Criteria 1 – The use of labeled data approximations to build, train, validate, or test a Supervised Learning model (e.g., labeled data approximations can come from synthesized and/or synthetic data).

Criteria 2 – Real labeled data samples,data from simulation environments , or synthetically generated samples are used to build, train, validate, or test a Supervised Learning model (e.g., real labeled data samples to develop Machine Learning model).

Should the item meet at least one of the criteria, then it is considered to be a ML-based item, and the following assumptions apply. The ML-based item deployment is assumed to be frozen (i.e., non-adaptive), meaning that the parameters do not adapt when fielded. Further the ML-based item model type is currently assumed to be of Supervised Learning type, as unsupervised learning, reinforcement learning, and generative AI, or other types of Machine Learning are not yet addressed in the following document. In addition, the following Machine Learning guidance assumes the ML-based item performance assurance, development assurance, and operational design domain have been identified and defined in the System/Subsystem Specification (SSS) and System/Subsystem Design Document (SSDD). An ML-based item which meets the stated criteria but does not meet the stated assumptions will be treated on a case-by-case basis.

Section 1.1 Purpose

This document contains the modifications and additions to traditional airworthiness guidance framework, processes, objectives, activities, explanatory text, and software item lifecycle data that should be addressed when ML data-driven algorithms are used. These additions address the use of data sets, the development, and training of the ML algorithm, traceability from data sets and models to requirements, and the verification evidence for them. Therefore, this document also applies to the data sets collected in the systems process that are used to define ML model behavior, ML data requirements, and/or ML architecture.

Section 1.2 Scope

Most programs surveyed, that are at a reasonably high Technology Readiness Level (TRL), use data-driven training techniques, i.e., Machine Learning, so that is the focus of this document. This document discusses the use of ML lifecycle, which consists of ML development lifecycle (MLDL) and ML implementation lifecycle (MLIL), for the development of ML data-driven algorithms in accordance with airworthiness standards. The scope of this document is limited to supervised learning, and frozen non-adaptive learning, especially for high development assurance level applications. Supervised learning, and frozen/non-adaptive are selected and considered because of the maturity and lower risk associated with these approaches.

Certification considerations for Unsupervised Learning, Reinforcement Learning, and Generative AI are inadequately discussed in this document, so follow-on work will delve into the applicability of the proposed assurance practices to those learning techniques. ML data-driven techniques in this document are selected based on their maturity and applicability. For non-supervised, generative AI, and adaptive systems, the processes required to establish assurance for their use and application in flight-critical applications are expected to be onerous. At this time, programs that need airworthiness guidance for adaptive, unsupervised learning, and/or generative AI are treated on a case-by-case basis. Only a brief but inadequate mention of some of those concepts is provided in Annex E-1.

The scope of this document primarily focuses on the development assurance, i.e., processes, objectives, activities, and output data items, associated with the ML lifecycle, which will primarily involve the ML data set assurance and ML learning assurance processes, objectives, activities, and output data items. Unique concerns associated with the hardware contributions to the ML lifecycle are not identified in the development of this document. Follow-on work will explore potential unique hardware concerns for the ML lifecycle, and, if necessary, propose applicable processes, objectives, activities, and output data items associated due to the hardware impact on the ML lifecycle. If vendors are aware of the hardware impacts to the ML lifecycle, they should be prepared to disclose those to, and coordinate those with, the airworthiness authorities.

The scope of the ML lifecycle in this document is limited to the development of the ML-based item to the appropriate development assurance level. Upon the completion of the development of the ML-based item to the appropriate development assurance and requirements, it is provided to the subsystem and/or system for integration, and ultimately fielding. The scope of this specification does not extend to those subsequent processes; however, those processes may also need to be appropriately updated to address the unique concerns associated with ML. That is, concepts like performance assurance, continuous monitoring, mitigations, and other subsystem and/or system techniques necessary for the safe integration of ML in the subsystem and/or system are beyond the scope of this document. Only a brief, but inadequate mention of some of those concepts is provided in Section 12. However, due to the statistical and probabilistic nature of ML, development assurance alone will not be enough, but instead those other concepts, more than likely, will also be necessary, i.e., performance assurance, continuous monitoring, mitigations, and other subsystem and/or system techniques may also be necessary for the safe integration of ML in the subsystem and/or system.

The scope of this document is limited to the development assurance techniques necessary for ML-based item and does not address the performance assurance requirements necessary for the subsystem and/or system. The development assurance techniques provide confidence that industry best practices for the production of the ML-based item are being followed to ensure that unintended, undesired, and unexpected behaviors are not present while required performance is present. A specific failure rate nor performance assurance rate is not assured by the development assurance technique, but instead a confidence can be provided. Further evaluations of the subsystem and/or system are necessary to ensure the ML-based item performance is adequate.

While important to the assurance argument for the ML-based item, the cyber considerations for the ML model, e.g., adversarial attacks, are beyond the scope of this document. While out of scope for this document, the cyber considerations are expected to work in concert with the airworthiness considerations during the ML lifecycle, and vice versa. Appropriate coordination between the cyber considerations and the airworthiness considerations should occur to deconflict any contradictory requirements.

Section 1.3 Relationship to Other Documents

The goal of this document is to serve as a means of compliance (MOC) for the development of ML-based software items for the U.S. Army Military Airworthiness Certification Criteria (AMACC). As shown in Figure 1, this document integrates with the current Army airworthiness framework. The AMACC uses industry standards to the greatest extent possible for MOCs. In the absence of MOC for ML-based software items, this document serves that role. In keeping with the desire of the AMACC to align with

industry standards for MOC, this document does the same by aligning with the direction of industry for software item assurance and ML-based item assurance.

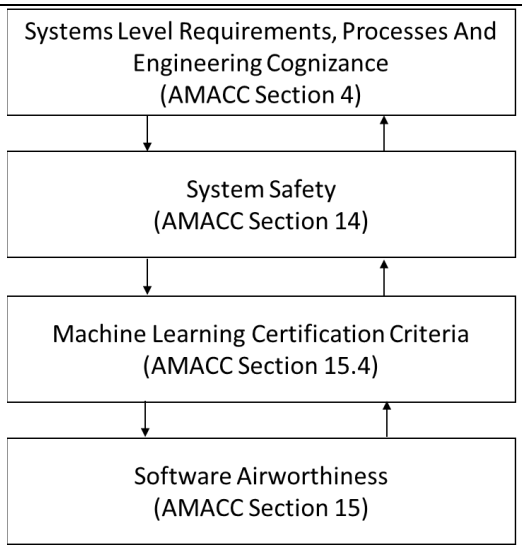

Figure 1. Guidelines covering development.

This document establishes the ML lifecycle, which includes the Machine Learning Development Lifecycle (MLDL) and ML Implementation Lifecycle (MLIL), which is not a standalone process. Instead, the ML lifecycle is a hybrid assurance process augmented by existing airworthiness guidance, e.g., SAE Aerospace Recommended Practice (ARP) 4754[9], SAE ARP 4761[10], RTCA DO-178C[11], etc. For ML data-driven algorithms, that canon of literature should be followed as well as the guidance provided herein. The reader is assumed to be familiar with that canon of airworthiness literature. However, where necessary, e.g., to highlight the similarity to existing guidance, excerpts from the canon will be referenced, especially from RTCA DO-178C. Those references are used to provide additional emphasis on the importance of those foundational materials. For the guidance within this document to be fully effective, the program desiring airworthiness approval should also follow the guidance provided in those foundational airworthiness guides, as those serve as a prerequisite to the guidance provided in this document.

In general, the MLDL captures the addition of the data science discipline to the usual system, safety, hardware, and software disciplines. The MLIL, meanwhile, is a continuation, with some minor enhancements, of the existing traditional software development assurance approach.

[9] Certification Considerations for Highly-Integrated Or Complex Aircraft Systems, SAE ARP 4754, 11/01/1996, [URL: https://www.sae.org/standards/content/arp4754/, Accessed: 12/05/2022]
[10] Guidelines And Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, SAE ARP 4761, 12/01/1996, [URL: https://www.sae.org/standards/content/arp4754/, Accessed: 12/05/2022]
[11] Software Considerations in Airborne Systems and Equipment Certification, RTCA DO-178C, 12/13/2011, [URL: https://my.rtca.org/productdetails?id=a1B36000001IcmwEAC, Accessed: 12/05/2022]

Other cornerstone aviation application standards and concepts that ML data-driven programs should be familiar with are covered in the following:

- Real-Time Operating Systems (RTOS) and Partitioning – Aeronautical Radio, Incorporated (ARINC) 653-ARINC, Aviation Application Software Standard Interface
    - 653P0-3 Avionics Application Software Standard Interface, Part 0, Overview of ARINC 653
    - 653P1-5 Avionics Application Software Standard Interface, Part 1, Required Services
    - 653P2-4 Avionics Application Software Standard Interface, Part 2, Extended Services
    - 653P3A-2 Avionics Application Software Standard Interface, Part 3A, Conformity Test Specifications for ARINC 653 Required Services
    - 653P3Bc1 Avionics Application Software Standard Interface, Part 3B, Conformity Test Specifications for ARINC 653 Extended Services
    - 653P4 Avionics Application Software Standard, Standard Interface, Part 4, Subset Services
    - 653P5-1 Avionics Application Software Standard Interface, Part 5, Core Software Recommended Capabilities
- Multi-core – Federal Aviation Administration (FAA) AC 20-193 – Use of Multi-Core Processors
- Failure Modes and Effects Analysis (FMEA) – SAE ARP 5580 Recommended Failure Modes and Effects Analysis (FMEA) Practices for Non-Automobile Applications

Section 1.3.1 Relationship to the AMACC

This document serves as the MOC for AMACC Section 15.4 This document offers the objectives, activities, and output data items required for the development of the ML-based item for flight critical applications. This document expects that the guidance of the AMACC is followed to establish the ML-based item requirements, appropriate development assurance level, identify the necessary ODD, and provide the necessary safety assessment details. For traditional software item development, the AMACC relies on appropriate industry standards to guide the development of those artifacts, e.g., SAE ARP 4754, SAE ARP 4761, RTCA DO-178, etc. To the greatest extent possible this document similarly relies on those on those industry standards. However, in some cases those standards do not provide adequate guidance, and additional guidance is needed. For example, in the area of Data Hazard Assessment, new guidance is necessary, so this document provides introductory expectations, while the DoD Joint Weapon Safety Working Group (JWSWG) is developing additional guidance.

Section 1.3.2 Relationship to SAE ARP 4754 Guidelines for Development of Civil Aircraft and Systems

FAA Advisory Circular (AC) 20-174 Development of Civil Aircraft and Systems recognizes SAE ARP 4754 Guidelines for Development of Civil Aircraft and Systems, "as an acceptable method for establishing a development assurance process." [FAA AC 20-174[12]] This process decomposes the systems requirements down to the subsystems, where subsystems are composed of items. For the purposes of this paper a software item is defined as a group of software elements that are treated as a single configuration item. The hardware item is not addressed in this document. The formal definition is

[12] FAA Advisory Circular (AC) 20-174 - Development of Civil Aircraft and Systems, September 30, 2011, [URL: https://www.faa.gov/regulations_policies/advisory_circulars/index.cfm/go/document.information/documentID/1019527, Accessed: 12/06/2022]

provided in SAE ARP 4754. Thus, an ML-based item is a group of software elements, which include an ML model, that are treated as a single configuration item.

The decomposition of the functional and item requirements and safety assessment requirements from the system to the ML-based software item will follow the processes detailed in SAE ARP 4754. While SAE ARP 4754 does not specifically address ML-based software item, its treatment of the functional and item requirement and safety requirement decomposition for traditional software is still applicable to the assignment of the same for ML-based software items.

The resultant decomposition output from following through the MLDL and the ML implementation lifecycle (MLIL) is an ML-based software item, i.e., a frozen ML inference model. That resultant ML-based software item will be combined with other ML-based software items, ML hardware items, traditional software items, or hardware items to build up an ML-based subsystem.

An example of an ML-based systems requirement is the following: (1) the ML-based system should include continuous monitoring of the input to the ML-based system and the output from the ML-based systems, (2) the ML-based system continuous input monitoring should ensure the input data is as expected for the operational design domain, and the output is within the safety thresholds for the ML-based system.

Two considerations not covered explicitly in SAE ARP 4754, but potentially necessary for an ML-based system, will be briefly discussed in the following sections. These two considerations are autonomy classification, which includes the treatment and considerations of autonomy, and the operational design domain, which covers defining the requirements for the operational context of the ML algorithm.

Section 1.3.2.1 Autonomy Classification

Reconciling the impact of autonomous behaviors on the modulation of the development assurance levels (DALs) is still being actively worked. The US Army Combat Capabilities Development Command (DEVCOM) Aviation & Missile Center (AvMC) Systems Readiness Directorate (SRD) is working on the inclusion of autonomy classifications assessments within the system safety assessment process. Army Military Airworthiness Certification Criteria (AMACC) section 9 and AMACC Appendix A offers guidance for the classification of autonomy based on level, type, and severity. This document looks to the guidance provided there and guidance in others[13,14] to begin assessing the impact of autonomous behaviors on the systems requirements and systems safety processes. Understandably a modulation of DAL will occur based on autonomy classification, but the exact relationship of that impact is still being investigated.

For this document, the concepts of autonomy and ML algorithm are kept separate. That is, autonomy can be enabled without the use of ML techniques; and the use of ML techniques can occur without evoking the existence of autonomy. This document focuses on establishing the assurance practices necessary for developing ML data-driven algorithms used in flight-critical applications, regardless of the level of autonomy.

[13] UL4600, American National Standards Institute (ANSI)/UL 4600: Standard for Safety for the Evaluation of Autonomous Products.
[14] Classifying AI Systems, Catherine Aiken, November 2021. [URL: https://cset.georgetown.edu/publication/classifying-ai-systems/]

Section 1.3.2.2 Operational Design Domain (ODD)

For ML data-driven algorithms, additional emphasis is necessary within SAE ARP 4754 and related on explicitly and completely defining the operational design domain (ODD). The ODD can be defined as "The set of environments and situations the item is intended to operate within. This includes not only direct environmental conditions and geographic restrictions, but also a characterization of the set of objects, events, and other conditions that will occur within that environment. NOTE: A system has a single ODD by definition. Assessment is made with regard to the entire ODD." [UL4600] UL4600 has a greater treatment of ODD than is presented here that is worth considering.

Civilian applications of the ODD consider robustness cases, which is where the deployed model receives data that violates the training data set to cause undesired behavior of the ML mode. For military applications, inclusion of robustness cases in the ODD is essential. Moreover, for military applications ODD should leverage, where possible, the Operational Mode Summary and Mission Profile, which are typical Reliability, Availability, and Maintainability products.

Emphasis should be placed on the systems design and decomposition phases to consider mitigating robustness conditions. In addition, considerations should be given to signal and source failures, feature and attribute dropouts, and missing and corrupted data.

Section 1.3.3 Relationship to SAE ARP 4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment

SAE ARP 4761 "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment" provides guidelines for conducting a safety assessment of the system and subsystem functional and item requirements identified in the SAE ARP 4754 process. This safety process outlined in SAE ARP 4761 consists of a traditional system, subsystem, software item, hardware item, and safety assessment tools: functional hazard assessment (FHA), preliminary system safety assessment (PSSA), and system safety assessment (SSA).

Those techniques worked reasonably well for traditional systems, but for ML-based systems those safety-assessment techniques may not be entirely adequate. Instead, new novel safety assessment techniques may be necessary to compensate where the traditional safety assessment techniques fall short. Examples of new techniques could include: STPA (Systems-Theoretic Process Analysis)/ STAMP (Systems-Theoretic Accident Model and Processes), autonomy classification techniques, data hazard assessment approaches, Bayesian Belief Networks (BBN), and Quantitative Safety Assessment (QSA). Those approaches may need to be used in parallel with the traditional safety assessment techniques. Being used in parallel will ensure that probabilistic based functions, e.g., ML, are adequately accounted for in the safety assessment. Traditional safety assessment techniques do not explicitly account for software failures, especially those that include a probabilistic component.

Given the traditional system focus of SAE ARP 4761, data hazard analysis techniques are absent. With ML's dependency upon data sets, hazard assessment of data source, data integrity, and data hazard implications are encouraged. Since data hazard assessment is a new safety assessment method, further elaboration of this method will occur when data governance is discussed.

The safety assessment process must consider the loss or malfunction of the ML-based subsystem and the effects on the system. Those considerations should drive the ML-based system requirements and safety requirements to be detected, protected, and recovered from ML-based subsystem malfunctions

and inadequacies. Possible protection and recovery mechanisms involve the use of monitors, mitigations, interlocks, and run-time assurance. Safety assessment may result in the safety requirements for monitors, mitigations, interlocks, or run-time assurance to mitigate the malfunction of the ML-based subsystem. The application of monitors, mitigations, interlocks, and run-time assurance does not discount the need for applying development assurance principles and practices.

Section 1.3.3.1 Development Assurance Level (DAL)

Three important outputs of the safety assessment process are the target development assurance level for functions (FDALs), target development assurance level for items (IDALs), and safety assessment requirements. FDAL, as the name implies, is based on the functional hazard assessment (FHA), fault tree analysis (FTA), and failure modes and effects analysis (FMEA). The output of those hazard assessments is an FDAL for functions. Functions are executed within items, i.e., items aggregate functions to be executed, so items are an aggregation of the highest FDAL they contain. IDAL is equal to the highest FDAL contained within the item. Partitioning, as described in ARINC 653, can be used to help isolate lower FDALs from higher FDALS. Using partitioning to isolate lower FDALs from higher FDALs is an architecting principle that increases robustness and leads to cost savings.

In general, the number of processes, objectives, and activities necessary at the systems, software, and complex electronics level is proportional to the DAL assessed for each, so a greater number of expected processes, objectives, and activities should be accomplished for higher DAL. The DAL is established at the systems and subsystems level based on the hazard assessment and functional decomposition. Discussed extensively in SAE ARP 4754 and SAE 4761, the DAL is assigned based on the top-level failure condition classification severity of the function, where DAL A is the highest and associated with a Catastrophic failure condition of the function.

Depending on the program, the DAL definitions may need to be harmonized with those provided in MIL-STD-882E.

For unmanned systems, the DAL definitions should be appropriately modulated in accordance with AR70-62 Unmanned Aircraft System (UAS) Airworthiness Qualification Levels. Airworthiness and safety organizations indicated that for UAS, third-party effects must be part of the airworthiness and safety consideration. Third-party performance and development assurance considerations can be determined by considering the design criteria for casualty expectation for ground-based population effects and the sense-and-avoid risk ratio for near mid-air collision effects. Precedent was established by the Army's Ground Based Sense and Avoid System, i.e., Ground Based Sense and Avoid System determined and met the development assurance and probabilistic performance goal. A similar course could be pursued to certify ML probabilistic-based systems. That is, ML systems could be developed to meet appropriate development assurance and probabilistic performance goals. This document primarily provides guidance for the development assurance goals, e.g., objectives and activities, for ML and how to achieve them. Follow-on work is necessary to determine and provide the approach to establish probabilistic performance requirements for ML and how to verify they are accomplished.

Section 1.3.4 Relationship to DO-178C and Supplements

FAA AC 20-115D[15] recognizes RTCA DO-178C, "is an acceptable means of compliance for the software aspects of type certification or Technical Standard Order (TSO) authorization." [FAA AC 20-115D] For the ML lifecycle, which consists of the ML development lifecycle (MLDL) and the ML implementation lifecycle (MLIL), where possible the guidance provided in DO-178C, and supplements was followed. Key processes, objectives, and activities that were leveraged from DO-178C include the planning process, verification process, configuration management process, quality assurance process, and certification liaison process.

The MLDL is a preliminary development lifecycle that outputs the ML requirements, ML model description, and ML data processing, which is to be used as equivalent to develop MLIL requirements. That is, the MLDL output of the ML requirements, supported by the ML model description and ML data processing description, can be used to enable the development of MLIL requirements, e.g., high-level and low-level requirements. Within the MLIL, the high-level requirements are used appropriately to decompose the input MLDL requirements, ML data requirements, and ML model requirements, and low-level requirements are used to develop the source code without additional details. Within the MLDL, new processes were introduced to address the unique assurance aspects associated with data assurance and training assurance. For those newly added MLDL processes, the structure and approach of DO-178C was leveraged.

The MLIL receives the ML requirements, ML model description and ML data processing description, and implements them as an ML inference model. MLIL directly uses DO-178C software lifecycle processes, with only minor modifications. Those modifications address the transition from the MLDL to the MLIL. In general, for the MLIL, the remaining DO-178C software development and verification processes remain unaffected by the ML data-driven algorithm implementation. The output of the DO-178C MLIL is an ML inference model with supporting traditional software elements, which will serve as the ML-based item. The ML-based item is equivalent to the software item.

This document is intended to be used with DO-178C. This document strives to align the precedent established by DO-178C to build an assurance case for the traditional software lifecycle, and account for the unique assurance needs for the data-driven ML lifecycle. The approach aligns with the structure and intent established in DO-178C while introducing new and novel assurance measures to address ML concerns. This approach seeks to minimize unfamiliar assurance terms and concepts by keeping as true as possible to the approaches taken in DO-178C and supplements. That means the structure of this document is similar to that of DO-178C, and where possible, sections from DO-178C are referenced. Moreover, where the guidance from DO-178C applies unchanged, that is indicated. Due to RTCA copyright concerns minimal quotations are included but instead references to DO-178C are used.

For the reader familiar with DO-178C the following should be noted with respect to the organization of this document:

[15] FAA AC 20-115D - Airborne Software Development Assurance Using EUROCAE ED-12( ) and RTCA DO-178( ), [URL: https://www.faa.gov/regulations_policies/advisory_circulars/index.cfm/go/document.information/documentID/1032046, Accessed: 12/06/2022].

a. As with DO-178C, Section 1 of this document contains explanatory text to aid the reader in understanding ML development and verification and therefore is intended to be informative.
b. Sections 2 through 11 of this document are adjusted to account for the additional ML lifecycle processes not covered in DO-178C. Where possible section titles, process, objectives, activities, and output data items were kept similar to original DO-178C traditional software items of the same name, but the reader should remember these are targeted at the data-driven ML lifecycle.
c. Annex A of this document provides objective, activity, and output data item tables modulated for development assurance level that are revised to align with the MLDL and MLIL.
d. The ML-based software item will most likely not solely be composed of an ML model but will also contain the necessary supporting traditional developed software elements. Those traditional software elements within the ML-based software item will be developed in accordance with the traditionally developed software process, e.g., developed in accordance with DO-178C and supplements.
e. ML-based software items will not operate within an ML-based system alone but will be enabled by traditional software items. That is, traditional software and/or traditional hardware items without ML will make up the majority of systems, where ML will only be a small fraction of the software and/or hardware on a system. Therefore, guidance from DO-178C should be used for all aspects of the traditional software lifecycle and DO-178C should be used for the processes, objectives, and activities of the MLIL. All guidance specific to ML development and verification is contained within this document.
f. The relationship of this document to the DO-178C Supplements is discussed in the following subsections.

Section 1.3.4.1 Relationship to DO-330 Tool Qualification Supplement

For the ML lifecycle, the use of tools may be extensive, e.g., the use of tools may range from the creation and processing of data to the creation and training of the ML model to the use of tools in the verification environment for ML model and data saliency.

When the applicant plans to use tools during the ML lifecycle, the applicant should comply with RTCA DO-178C and RTCA DO-330[16]. Tool qualification is necessary for a tool when a process, objective, or activity is automated by a tool and the output of the tool is not verified. When tool qualification is necessary the DO-330 tool qualification objectives must be appropriately followed.

For ML, neither DO-178C nor DO-330 provides specific guidance for tool qualification for ML-based tools, e.g., Large Language Model (LLM) based tools. The applicant should coordinate any use of ML-based tools with the certifying authority. The traditional tools used during the ML lifecycle, i.e., the MLDL and ML Implementation Lifecycle (MLIL), should conform to the current tool qualification guidance provided in DO-178C and DO-330.

Due to the potential simulation-to-real gap consequences, tool qualification and verification, validation, and accreditation (VV&A) is strongly advisable for tools used to create the synthetic data set used for

[16] RTCA DO-330, Software Tool Qualification Considerations, 12/13/2011, [URL: https://my.rtca.org/productdetails?id=a1B36000001IcfkEAC, Accessed: 12/06/2022]

training, validating, and testing the supervised learning model[17]. The same is recommended, i.e., tool qualification and VV&A, for simulation environments used for the development of ML models.

Section 1.3.4.2 Relationship to DO-331 Model-Based Development

Within RTCA DO-331[18], there are two levels of models: (1) specification model and (2) design model. For DO-331, the specification model is equivalent to the high-level requirements, and the design model is equivalent to the low-level requirements. DO-331 is explicit that the specification model and design model cannot be the same. Neither of these DO-331 concepts is fully aligned with the concept of the ML model, nor, at the time of its creation in 2011, was DO-331 envisioned to cover ML models. That is, while the term model is used in both contexts, the intent of the concepts is unique.

The objectives of DO-331 would only apply to the MLDL or MLIL (ML implementation lifecycle) if Model-Based Development (MBD) techniques, in accordance with DO-331, are used to develop ML-based software item. For example, if the MBD specification model is used to create the ML data set or ML model during the MLDL, then guidance from DO-331 should be followed in addition to the guidance provided in this document. The assurance practices presented in this document do not explicitly cover or consider the processes, objectives, or activities that may be unique to the use of MBD within the MLDL or the MLIL, nor does it preclude the use of such techniques. Follow-on efforts will further investigate if additional unique MBD processes, objectives, or activities are necessary should MBD be used during the MLDL or MLIL.

Section 1.3.4.3 Relationship to DO-332 Object Oriented Technology

The RTCA DO-332[19] supplement will be most applicable to the ML implementation lifecycle, and not to the MLDL. The MLDL allows for the use of interpreted programming languages for ML model development. Interpreted programming languages, e.g., Python[20], have not been shown to be compliant with DO-332, so are not applicable for the MLIL. Alternative compliant programming languages should be considered for MLIL, e.g., compiled programming languages.

One possible course of action is that the implementor, in the MLIL, will transition to a programming language that has been shown to be compliant with DO-332, e.g., C++ (ISO/IEC 14882:2020). Based on architectural decisions, there may be exceptions to this approach, but those would need to be treated on a case-by-case basis, e.g., potential use of MBD in the MLDL or the MLIL. Additional considerations for the use of interpreted programming languages are provided in Annex E-4.

[17] Carter, G., Vinegar, C., Terres, V., and Rupert, J., "Considerations for Tool Qualification in Flight-Critical Applications using Machine Learning", 43rd Digital Avionics Systems Conference (DASC), San Diego, California, USA. OCT 2024.

[18] RTCA DO-331, Model Based Development and Verification Supplement to DO-178C and DO-278A, 12/13/2011, [URL: https://my.rtca.org/productdetails?id=a1B36000001IcfiEAC, Accessed: 12/06/2022]

[19] RTCA DO-332, Object Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A, 12/13/2011, [URL: https://my.rtca.org/productdetails?id=a1B36000001IcfgEAC, Accessed: 12/06/2022]

[20] H. Glenn Carter, Alexander Chan, Chris Vinegar, Jason Rupert, "Concerns with using Python in Machine Learning Flight Critical Applications". Presented at Vertical Flight Society (VFS) Forum 79 Conference, 2023, DOI 10.4050/F-0079-2023-18015.

Section 1.3.4.4 Relationship to DO-333 Formal Methods

RTCA DO-333[21] Formal methods, to the extent previously applicable to DO-178C, remain applicable to the ML lifecycle. Unfortunately, no significant developments have occurred to increase their applicability more than previously. That is, a literature search did not uncover any major civilian aviation, military aviation, or other applications of formal methods that successfully led to the airworthiness certification of ML. It appears that formal methods have not crossed the threshold of research and development to be applicable to programs in flight and safety-critical complex operation design domains for ML. Moreover, there are challenges with the "curse of dimensionality" posed by ML for formal methods-based approaches, i.e., the dimensionality of the ML model causes the complexity to increase beyond what the formal method tools are able to assess. Noteworthy is that neither SAE G-34 ARP 6983 nor Safety Critical Systems Club (SCSC) 153B make specific mention of using formal methods during the MLDL. Additional considerations for the use of formal methods to certify ML is provided in Annex E-2.

Section 1.3.5 Relationship to DO-254 Airborne Electronic Hardware

This document primarily focuses on developing ML-based software items; however, complex ML hardware items, i.e., items applicable to RTCA DO-254[22], were explored, but not discussed in detail. Specifically, an informal coordination meeting was held with a Software Architect at WindRiver to discuss complex ML hardware certification. His initial indications were that Tensor Processing Units (TPUs), like those under development by NVIDIA, will be easier to certify than multi-core central processing units (CPUs). Certification of TPUs was more likely due to the simpler, more focused functionality of TPUs than that of CPUs. Follow-on work is necessary to explore the certification of ML-based hardware items more fully, e.g., TPUs, and the certification of ML-based software items on multi-core Graphics Processing Unit (GPU) hardware or other application-specific purpose build hardware deployments. Some follow-on work occurred[23] but additional work is necessary to fully explore these impacts. Important to note, just as with ML-based software items, ML-based hardware items will most likely be a combination of traditional complex hardware elements and ML-specific complex hardware elements.

Section 1.3.6 Relationship to SAE ARP 6983 Recommended Practice for Development and Certification/Approval of Aeronautical Safety-Related Products Implementing ML

The SAE G-34 committee is creating the SAE ARP 6983[24], which is an aggregation of ML certification concepts from several companies and assurance organizations, e.g., European Union Aviation Safety Agency (EASA), Boeing, Airbus, Thales Group, etc. The goal of SAE ARP 6983 is to provide development assurance guidance for the safe integration of ML into civilian aviation applications. Authoring of this

[21] RTCA DO-333, Formal Methods Supplement to DO-178C and DO-278A, 12/13/2011, [URL: https://my.rtca.org/productdetails?id=a1B36000001IcfeEAC, Accessed: 12/06/2022]
[22] RTCA DO-254, Design Assurance Guidance for Airborne Electronic Hardware, 4/19/2000, [URL: https://my.rtca.org/productdetails?id=a1B36000001IcjTEAS, Accessed: 12/06/2022]
[23] Carter, H. G., Chan, A., Terres, V., Rupert, J., Scales, A., "Navigating Airworthiness Concerns with Deploying AI/ML Applications – A Brief Survey", Vertical Flight Society Forum 80 Conference 2024, Montreal, Canada, MAR 2024.
[24] Recommended Practice for Development and Certification/Approval of Aeronautical Safety-Related Products Implementing ML, SAE ARP MLDL Environment, SAE G-34, [URL: https://www.sae.org/standards/content/arp6983/, Accessed: 12/05/2022 – version available via committee membership]

document involved coordination with the SAE G-34 and participation in the creation of the SAE ARP 6983. Some of the concepts presented in this document originate from the concepts introduced in drafts of SAE ARP 6983, and from other publications that contributed key concepts to the production of SAE ARP 6983, i.e., EASA Roadmap, EASA Level 1 Guidance, and SCSC-153C. Guidance from many other organizations was also reviewed, e.g., FDA, NIST, NTSB, etc., but the guidance from SAE ARP 6983 was found to most closely align with the Army's existing aviation framework, i.e., AMACC and AR70-62, and the US military's aviation airworthiness framework, i.e., MIL-STD-516C.

SAE ARP 6983, being at draft status means several sections are left incomplete and many inconsistencies and open questions remain. As of August 2024, Draft 6 has yet to be released, but is anticipated prior to the mid of CY2025. Given this status, instead this document seeks to remain aligned with the overall intent and concepts of the current aviation community and framework. Where necessary deviations are taken to tailor this document specifically to the Army's current airworthiness needs.

Section 1.3.7 Relationship to EASA Roadmap and Level 1 and 2 Guidance

EASA funded the development of Concepts of Design Assurance for Neural Networks Part I & II[25,26] and authored several standards[27,28,29] to explore safely integrating ML into European civilian aviation airspace. Their guidance is taking a three-level pragmatic approach: Level 1 – Advisory, Level 2 – Collaboration, and Level 3 – Autonomous.

EASA levels provide a pragmatic growth path from the certification of Level 1 type application of ML, while working toward providing certification guidance of Level 3 type ML, i.e., the use of independent unsupervised nonoverridable AI technology in certified flight-critical applications. The EASA Roadmap provides the timeline EASA is working toward providing guidance for each level. Currently Level 1 and Level 2 guidance has been published. Moreover, their work only addresses supervised learning but plans to address reinforcement learning. A literature review of the EASA work, especially the Level 1 and 2 Guidance, was performed. For Level 1 and 2 EASA adds 60 unique objectives for ML applications. Finalization of the EASA ML guidance is anticipated to occur by 2029.

A review of the EASA objectives was conducted, and a report provided to the Systems Readiness Directorate AI Working Group (WG) for review and consideration. During the review, the SRD AW WG agreed with the intent of those objectives and believed they should be carried forward into ML airworthiness assurance standards. Therefore, where appropriate, this document incorporates the EASA

[25] Concepts of Design Assurance for Neural Networks Part I, EASA, March 31, 2020, [URL https://www.easa.europa.eu/sites/default/files/dfu/EASA-DDLN-Concepts-of-Design-Assurance-for-Neural-Networks-CoDANN.pdf, Accessed: 10/31/2022].
[26] Concepts of Design Assurance for Neural Networks Part II, EASA, May 19, 2021, [URL: https://www.easa.europa.eu/sites/default/files/dfu/ddln_easa_codann2_public.pdf, Accessed: 10/31/2022].
[27] EASA AI Roadmap: human-centric approach to AI in aviation, February 2020, Version 1.0, [URL: https://www.easa.europa.eu/sites/default/files/dfu/EASA-AI-Roadmap-v1.0.pdf, Accessed: 10/31/2022]
[28] EASA Concept Paper: First usable guidance for Level 1 machine learning applications, Proposed Issue 01, April 2021, [URL: https://www.easa.europa.eu/sites/default/files/dfu/easa_concept_paper_first_usable_guidance_for_level_1_machine_learning_applications_-_proposed_issue_01_1.pdf, Accessed: 10/31/2022]
[29] EASA Concept Paper: guidance for Level 1 & 2 machine learning applications, Issue 02, March 20224. [URL: https://www.easa.europa.eu/en/document-library/general-publications/easa-artificial-intelligence-concept-paper-issue-2#group-easa-downloads, Accessed: 8/26/2024]

objectives. Moreover, this document aligns with the current EASA assumption for frozen/locked/non-adaptive behavior when deployed.

Worth mentioning is that many of the EASA Level 1 and 2 objectives appear in SAE ARP 6983.

Section 1.3.8 Relationship to SCSC-153 Safety Assurance Objectives for Autonomous Systems
SCSC-153 Safety Assurance Objectives for Autonomous Systems[30] is a domain agnostic listing of objectives and activities for the safe application of autonomy concepts. While not specifically focusing on ML, many of the objectives and activities discussed in SCSC-153 apply to the safe inclusion of ML in safety-critical applications. A literature review of the SCSC-153 work was performed. The objectives and activities were provided to the SRD AI WG for review and consideration. Like the EASA Level 1 and 2 objectives, many of the SCSC-153 objectives were found to be of value and should be considered in any ML airworthiness assurance standard developed.

SCSC-153 began to work through the complexity of adaptive systems, and considerations associated with reinforcement learning as well as unsupervised learning. SCSC-153's treatment of adaptive learning has greatly influenced this document to strongly discourage its use in fielded systems. The treatment of reinforcement learning illustrated there is a path forward for its potential safe usage, albeit in a frozen state when deployed in an inference model.

Section 1.3.9 Relationship to DO-297 Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations
Deployments of ML may involve the combination of ML-based software items and ML-based complex hardware items, e.g., ML model executed via GPU hardware. While details and approaches are still being resolved, the deployment of ML-based software elements/items combined with ML-based complex hardware elements/items may be able to be managed similar to that described in RTCA DO-297 Integrated Modular Avionics (IMA)[31]. That is, the necessary software and complex hardware elements/items could be combined as follows:

- ML-based software element (e.g., neural network) plus traditional software element produces an ML-based software item, which is equal to an ML-based software component.
- ML-based complex hardware element (e.g., GPU) plus traditional hardware element produces an ML-based hardware item, which is equal to an ML-based hardware component.

Then, the following could be proposed:

- ML-based software component plus ML-based complex hardware component produces an ML-based module.

In cases where such combinations of ML-based software item and ML-based complex hardware item are used to build line replaceable unit and IMA behavior, an aircraft level IMA certification plan and verification and validation plan will need to be developed along with an associated system level IMA certification plan and verification and validation plan. In addition to those plans each of the

[30] Safety Assurance Objectives for Autonomous Systems, Safety-critical Systems Club (SCSC) 153, [URL: https://scsc.uk/r153B:1?t=1, Accessed: 12/06/2022]
[31] RTCA DO-297, Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations, 11/8/2005, [URL: https://my.rtca.org/productdetails?id=a1B36000001IchFEAS, Accessed: 12/06/2022]

components included in the modules will need accompanying certification plans, accomplishment summaries, and supporting output verification and validation data items.

Follow-on work will thoroughly examine the build-up of IMAs from ML-based items. This document primarily focuses on the processes, objectives, activities, and output data items necessary for ML-based software items.

Section 1.4 Document Overview

This document is intended to be used in conjunction with SAE ARP 4754, SAE ARP 4761, RTCA DO-178C and appropriate DO-178C supplements. An overview of the document is provided in Figure 2. By design, the layout of this document is similar to DO-178C and aims to be easily followed by those familiar with DO-178C. Appropriate modifications and additions were necessary to Sections 2 through 12, and Annex A provides support for the unique attributes of ML.

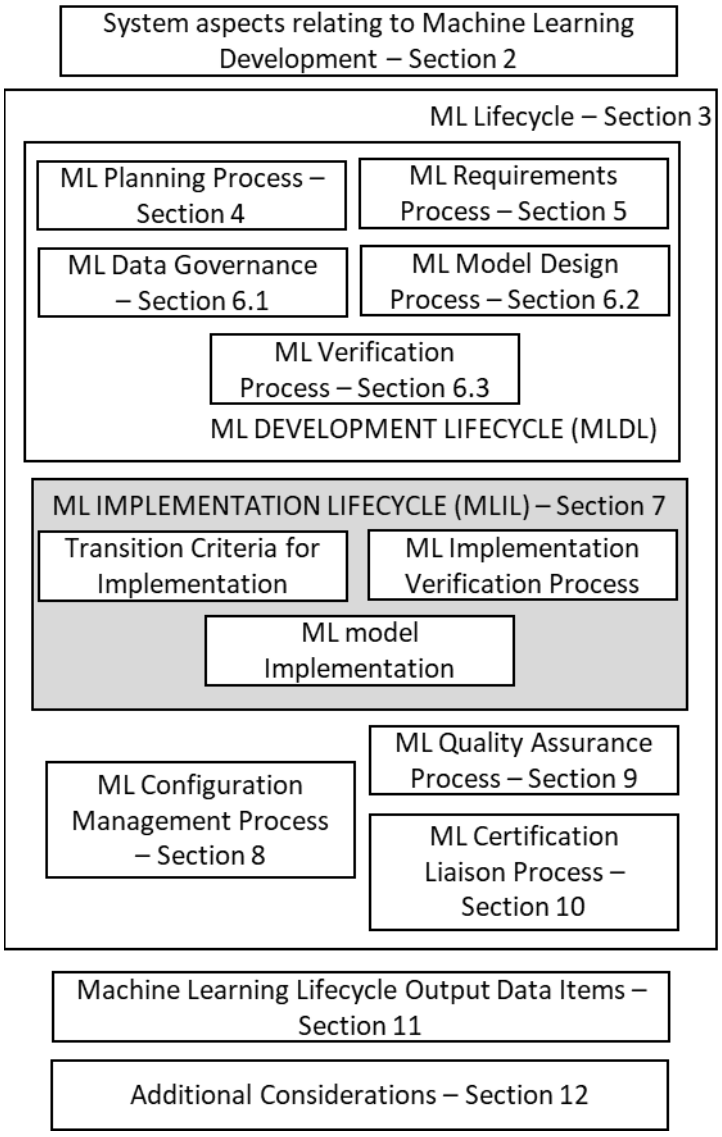

Figure 2. Document content overview.

Traditional software item development assurance processes, objectives, and activities are shown with Grey Fill, while the ML-based items discussed in this document are shown with white fill.

Section 1.4.1 How to Use This Document

The following approach should be taken to maximize the benefit of this document:

a. This document is applicable within the current existing aviation framework guidance, e.g., SAE ARP 4754, SAE ARP 4761, RTCA DO-178C, and DO-178C supplements. This document can be used stand-alone to understand the Machine Learning Development Lifecycle expectations, but the greatest benefit of this document is achieved when it is used in conjunction with software lifecycle item assurance development guidance, e.g., DO-178C and DO-178C supplements.

b. Within this document "shall" statements are not used, but instead, the appropriate and applicable processes, objectives, activities, and output data items are mandatory. Annex A provides tables that show the appropriate and applicable objectives and activities. That is, Annex A presents a modulation of objectives and activities per development assurance level (DAL), so that they are reduced for lower DAL. The annex should not be used as a checklist without following the processes, objective, activities, and output data item presented in the rest of this document.
c. Substantiation of accomplishing the necessary objectives and activities is provided through the production of output data items. The recommended content of the output data items discussed through the text of this document, and Section 11 presents the overall listing of all recommended output data items and their minimally required content. Annex A also provides a mapping of the output data item to the objectives and activities.
d. The vendor developing the ML-based software item should have the appropriate ML-based system requirements, e.g., performance assurance expectations, definition of the operational design domain, and system safety artifacts prior to beginning the ML-based software item development. Those requirements should be provided to the ML-based item from the systems decomposition process. Successful deployment of the ML-based software item is dependent upon the supporting ML-based system infrastructure. Section 12 discusses a few other ML-based system considerations.

Section 1.5 Characteristics of ML Development and Verification

An ML model combines the ML algorithm, and the data set used to train, validate, and test the ML algorithm. As part of the ML development and verification process, data sets should represent the operational design domain (ODD) and operational use scenarios. The data sets used to train, validate, and test should have appropriate independence between portions of the data set.

The ML algorithms are characterized by the learning technique that is used, e.g., convolutional neural network. The techniques used should be suitable, while the data set should be at the appropriate level of abstraction and format to be processed by the ML algorithm. The ML Data Set Standard and ML Model Standard are the means to describe the data assurance standard techniques and ML algorithm standards techniques. Those standards should be used to help develop data sets and ML models that are fit for purpose.

Section 1.5.1 ML algorithm

Within this document, the ML algorithm is the mathematical representation, which will be used during the training process. The ML algorithm mathematical representation may include the number of neurons, layers, or specific technique, e.g., linear regression, random forest, gradient boosting, Q-learning, Proximal Policy Optimization Algorithms (PPO), etc. The basic algorithm is devoid of a priori values, where those values are determined through an iterative training process for a specific set of training data. In general, in the context of this document, ML algorithms are data-driven, so are only as good as the data used to train them. Moreover, the ML algorithm complexity, i.e., number of weights and biases, can be structured such that the algorithm will overfit the training data set. Overfitting is a case where the algorithm will generalize poorly on test data and on deployed data (e.g., data received while in the field) but performs well on training data. For supervised learning, there are a number of methods to address overfitting, e.g., regularization, obtaining more data and simplifying complexity by

reducing the neurons. There are other benefits to reducing the number of neurons, e.g., model execution time, reduced training time, reduced computing power. Reducing complexity too much can have the unintended consequence of underfitting, which is where the model does not generalize well to training data nor on test, verification, and deployed data. Independent training, validation, and test data sets, i.e., simulation environments and scenarios, can help reduce the occurrence of overfitting. Independent training, validation and test simulation environments and scenarios also helps to reduce the possibility of catastrophic forgetting.

While the terminology and discussion within this document tends to be more biased toward neural network type ML algorithms, no one ML algorithm type, as long as it is supervised learning, is preferred over another ML algorithm.

Within the MLDL Planning Process, the ML Model Development Plan outlines the planned approach for designing and developing the ML algorithm, which will be based on the allocated ML Model requirements. Within the ML Model Design Process, the ML algorithm design will be captured and finalized in the ML Model Description.

Section 1.5.2 Data Set Requirements

For the ML lifecycle, the requirements from which the data set is developed should be identified. Requirements from which the data set is developed should be external to the data set, e.g., ML data set high-level requirements. Those requirements should provide details and constraints to enable data set development and verification activities and have appropriate traceability. For example, the ODD in the system and subsystem requirements will be used to develop the data set requirements.

Section 1.5.3 ML data sets

Within this document ML data sets, for supervised learning type ML algorithms, capture feature, attribute, sources, and signal characteristics, and, when necessary, capture dynamic temporal environments and scenarios. For each ML type, the data set can be divided into three independent sets:

- Training Data Set – used to train the ML model.
- Validation Data Set – used to validate and optimize the ML model hyperparameters, weights, and/or policy.
- Test Data Set – used to test the generalization and robustness of the ML model behavior on previously unseen data and facilitate ML model behavior explanation and verification.

For simulations and scenarios, the split into independent training, validation, and test data sets may be in the simulation environment confirmation, or the scenarios. By having independent data sets, simulation environments and scenarios, demonstrate that the ML model can adapt to unique independent configurations and transfer learning to new simulation environments and scenarios. Different simulation environments and scenarios for the training, validation and test phases also have the benefit of confirming overfitting and catastrophic forgetting is not occurring.

The data sets should have the appropriate number of scenarios. The scenarios should cover the operations within the ODD, where those operations include nominal and adverse conditions. The operations within the ODD should be detailed within the systems assessment and development process and allocated to the ML-based item.

For synthetic data sets, independence may also imply that unique tools were used to create the data set, or independent sources originated the data set. This independence will help to reduce the possibility of synthetic data bias and common mode failures across training, validation, and test data sets. These issues are due to the sim-to-real gap that can exist with synthetic data sets. Synthetic data is completely generated by simulation, whereas synthesized data is generated through the digital processing of "real" world data.

ML data sets include feature, attribute, source, and signal characteristics that drive the ML algorithm during training via the training data set. That is, the training data set is used to automatically produce parameter values, such as neural network weights and biases, during the ML training process. During the ML development lifecycle, how well the model performs is determined by evaluating the generalization of the trained model on test data set.

After the transition to the ML implementation lifecycle (MLIL), the test data set is used to ensure the semantics of the ML model design, from the MLDL, is preserved for the same data set inputs, e.g., the ML model performance and accuracy are preserved for the test data set.

The three types of data sets listed above, i.e., training, validation, and test data sets, are the minimum data sets expected, as other approaches may require the use of additional ML data sets. For example, using different unique data sets, ML components may be tested individually and then grouped together or with traditional software solutions for sub-system and system testing. In addition, an ML-based sub-system and/or system level hold-out data set may be used to appropriately test the ML-based sub-system and/or system. In such cases the ML-based sub-system and or ML-based system data set is called Sub-system/System Test Data Set(s). Such a data set would be addressed through the appropriate sub-system and system level planning processes, and account for the operational design domain.

Section 1.5.4 ML Element

ML elements are not able to function alone but instead are deployed with traditional hardware and/or software elements that support their execution. For example, the traditional elements are necessary to provide the inputs to, and receive the output from the ML element, i.e., the traditionally developed elements may be responsible for translating the data to the expected input format for the ML model. As shown in Figure 3 the grouping of those elements can be called an ML-based item, either hardware (HW) or software (SW). The ML-based item is combined with other ML-based items or traditional items to build up an ML-based subsystem. The ML-based subsystem is combined with other ML-based subsystems or other traditional subsystems to form the ML-based system.

This approach remains consistent with the existing aviation certification framework that involves the use of the following terms: system, subsystem, and item. Refer to aviation industry standards for the definition of those terms.

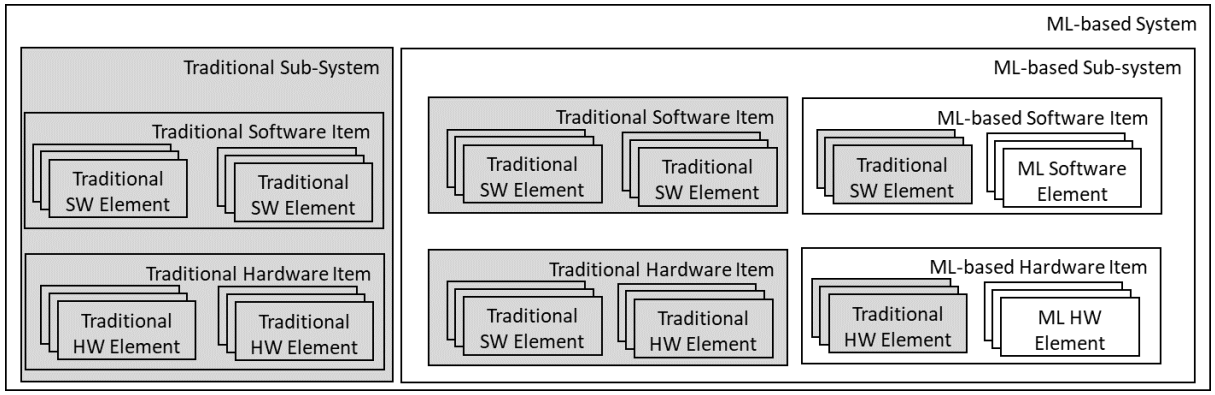

Figure 3. ML-Based system composition.

Notes for Figure 3:

- Note 1: Traditional software item and element will be developed in accordance with (IAW) DO-178C, while traditional hardware item and element will be developed IAW DO-254.
- Note 2: ML-based software item and elements will be developed IAW this document.
- Note 3: Traditional software item/element and traditional hardware items/element are shown with Grey Fill, while the ML-based items (software and hardware) and element are shown in white fill.
- Note 4: DO-178C uses the term software component. SAE ARP 4754 indicates that software item is equivalent to the concept of software component used in DO-178B and DO-178C. Within this document, an effort is made to use the term ML-based item, where that is assumed to be equivalent to the term component.
- Note 5: As with traditional items, ML-based software items may contain one or more ML elements, e.g., ML-based items may contain ensembles or algorithms composed of multiple ML algorithms.

The bulk of this document focuses on the processes, objectives, activities, and output data items necessary for the ML-based software item. This discussion of the ML element was included to acknowledge that for most fielded systems, the ML software element will be dependent upon a host of surrounding traditional software and hardware elements. The safe and airworthy inclusion of the ML-based software item requires appropriate architectural, design, and assurance considerations for the other traditional elements and items.

Any traditional elements within an ML-based item may be assigned safety requirements, which are developed during the system development and system safety assessment processes.

All elements within an item should be developed to the same development assurance level as assigned to the item. This is true for traditional items and remains true for ML-based items.

In general items can be thought of as being analogous to MIL-STD-498's concept of Computer Software Configuration Item.

The need for programs to follow the existing airworthiness guidance is important because ML data-driven algorithms do not stand alone, but instead are participatory items within a traditional system,

subsystems, and items. When ML elements are added to traditional items, they become ML-based items. Then a build-up occurs by creating ML-based subsystems when the ML-based item combines with traditional subsystems. Then an ML-based system is formed when the ML-based subsystem is added to the traditional system. The traditional items in these hybrid systems should follow the existing traditional airworthiness guidance.

Section 1.5.5 High-level and Low-level Requirements

Within this document, the terms "high-level" and "low-level" requirements are not used to describe the requirements within the MLDL. Instead, those terms are applicable to the process within the MLIL.

ML lifecycle is made up of two lifecycles: MLDL and MLIL.

Within the MLDL, as shown in Figure 4, there are three types of requirements:

- ML requirements – A decomposition of the system requirements allocated to the ML-based item.
- ML data requirements – A decomposition of the MLDL requirements into requirements for the ML data set.
- ML model requirement – A decomposition of the MLDL requirements for the ML model.

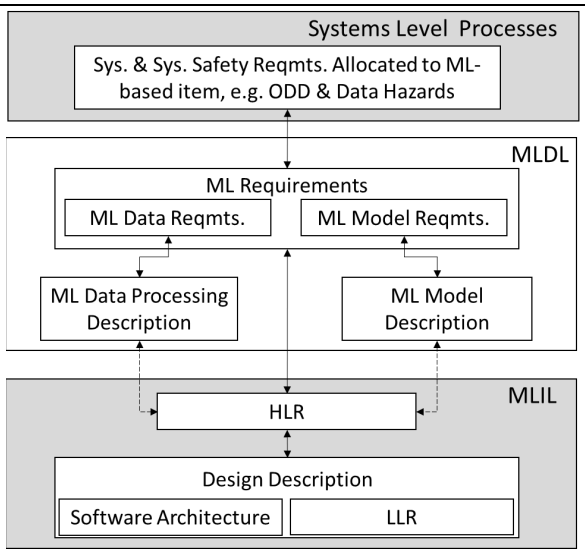

Figure 4. MLIL Requirements Process.

The output of the MLDL also includes the following two descriptions:

- ML data processing description – description of the data process used for the creation of the ML data set used for training, validation, and testing of the ML model.
- ML model description – description of the ML model that results from the ML model training, validation, and testing process.

MLDL requirements, ML Data Processing Description and ML model description serve as the input to the MLIL requirement process. Thus, there should be bi-directional traceability from the MLIL HLRs to the ML requirements, ML model description, and ML Data Processing Description.

Section 1.5.6 Recommendations and clarifications

The following are recommendations for ML-based item development, especially for high development assurance level ML-based items:

1. Continuous learning, which involves weights being dynamically updated in a deployed/fielded ML model, is not encouraged. Instead, locked/frozen deployment is recommended, where weights are static in the deployed/field ML model. Adaptive learning brings numerous challenges where assurance guidance is not sufficient to ensure guaranteed safe functionality in flight-critical applications.
2. Modular pipelines decompose the functions into small modules and elements, which are easier to test and debug, whereas large monolithic single element architectures are difficult to test and debug.
3. Use of an ML-based item is recommended to be advisory, i.e., Level 1, rather than on-the loop, i.e., Level 2, or in-the-loop, i.e., Level 3. More integrated and automated use of ML in flight-critical applications increases the safety concern. While ML concerns are still being mitigated, this document recommends a crawl-walk-run integration approach.
4. The system and subsystem requirements process should establish quantitative and qualitative requirements for the ML-based item, e.g., include probability, availability, integrity, and performance consideration.
5. System safety assessment process considers the ML function, defines ML quantitative and qualitative safety requirements, assess the ML ODD.
6. System and subsystem requirements incorporating ML-based items should incorporate appropriate monitor and mitigating strategies.
7. Appropriate tool qualification and/or VV&A should be applied to the training, validation, and test simulation environment and scenarios.
8. ML-based item testing should include extensive traditional and non-traditional robustness testing, e.g., 161 million encounter model test cases were executed for ACAS X. That is, non-traditional adverse stress testing verification approaches should be used to ensure the robustness of the ML algorithm to off-nominal, edge, and corner cases.
9. Only supervised learning type ML algorithms are encouraged for use in aviation applications. Data-driven training is the current focus of assurance communities, with specific focus on supervised learning. Additional assurance guidance for unsupervised learning, reinforcement learning, and Generative AI/Large Language Models (LLM) is being produced and will be provided in a follow-on work. Until that follow-on work is produced unsupervised learning and Generative AI ML-based items are not recommended for flight critical applications.
10. When possible, use of tabular solution methods is encouraged over the use of approximate solution methods. For tabular solution methods the complete state, action, and reward spaces can be explored, but for approximate solution methods that is not possible.
11. If approximate solution methods are used, then continuous monitoring and mitigations should be in place to restrict the ML algorithm to acceptably safety outputs.

These recommendations and the development assurance processes, guidance, activities, and output data items that follow address the concerns that exist with ML-based items, where those concerns are the following:

- Self-Modification: Includes online and adaptive learning, which undermines any previous training, testing, verification, and qualification.
- Sim-to-real: Gap between the simulated synthetic environment representation and reality.
- Distribution shift: Learning/training environment differs from the deployed environment, e.g., environment scalability (multi-agent) and heterogeneity concerns versus training environment.
- Catastrophic Forgetting: During retraining, neuron values are overwritten without retaining previously learned details.
- Interruptible: Monitoring or mitigation triggers the need to interrupt the ML model, e.g., vote out or not use.
- Value-alignment: ML model is not aligned with required values and includes negative side effects. Note: these are not adversarial AI concerns, which are cyber/IA concerns.

The following is a clarification between autonomy and ML. A clarification should be made between autonomy and ML. Autonomy is a desired function, whereas ML is a computer science algorithm development technique. Autonomy is a function of the system and can be of various types and levels, which are addressed in AMACC/Appendix A. ML is not necessary for autonomy to be present, and ML can be used where autonomous functions do not exist. The purpose of this document is not to address autonomy, its assessment, or its modulation of the development assurance levels. For guidance on that topic refer to AMACC/Appendix A, while follow-on work will more thoroughly address the necessary modulation of development assurance based on autonomy. Instead, the focus of this document is to provide the airworthiness certification criteria guidance for the use of ML in flight-critical applications, specifically the use of supervised learning.

Section 2 System Aspects Relating to ML Development

While the system lifecycle processes are detailed in foundational aviation industry guidance materials, e.g., SAE ARP 4754 and SAE ARP 4761, this section highlights aspects of the system lifecycle processes that are essential to successfully execute the ML lifecycle. The ML lifecycle includes the ML development lifecycle and the ML implementation lifecycle.

Discussed in this section are:

- System requirements allocation to ML-based item (see Section 2.1).
- The information flow between the system and ML lifecycle processes and between the ML and hardware lifecycle processes (see Section 2.2).
- The system safety assessment process, failure conditions, software level definitions, and ML level determination (see Section 2.3).
- Architectural considerations (see Section 2.4).
- ML considerations in system lifecycle processes (see Section 2.5).
- System considerations in ML lifecycle processes (see Section 2.6).

The term "system" in the context of this document refers to the airborne system and equipment only, not to the wider definition of a system that might include operators, operational procedures, etc.

Within the system decomposition process, justifications should be made for the use of ML. That is: "Given the current low maturity of autonomy-related techniques, there is a strong safety-related argument that wherever possible more traditional techniques should be used."[32] Bruce Nagy[33] makes a similar request that the use of ML technologies over traditional techniques should be justified to outweigh the potential risks and uncertainties associated with their performance: "Discuss and document a justification for the Proposed ML Algorithm's development application vs. a Traditional Code development to explain why an ML algorithm is a "better" fit to the function requirement."

Figure 5 provides a highlight of the information flow between the system and ML lifecycles, where the ML lifecycle includes two lifecycles: (1) ML Development Lifecycle (MLDL), and (2) ML Implementation Lifecycle (MLIL). The current version of this document focuses on the Software Development Lifecycle portion of the MLIL, where follow-on work will address the unique hardware assurance concerns associated with ML deployment, e.g., general purpose use of GPUs for ML and ML-specific chips. The first lifecycle, i.e., MLDL, takes the subsystem/system requirements and associated item DAL assignment allocated to the MLDL as inputs, and outputs at least the test dataset, the ML model description, and ML Data Processing Description. The second lifecycle, i.e., MLIL, takes as a minimum input the test data set, the ML model description, the ML Data Processing Description, and outputs the ML inference model. The MLIL (Software Development Lifecycle) primarily follows the guidance provided in DO-178C, with added processes to account for the implementation of the ML Model Description and ML inference model verification. Further details of the objectives and activities involved in each process of the ML lifecycle are covered below. Different types of data-driven ML techniques may require additional MLIL

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

[33] Fourteen Tips to Increase Confidence in the Performance of Artificial Intelligence (AI)/Machine Learning (ML) Functions during the Five Stages of Development, National Fire Control Symposium, 02/14/2022 to 02/17/2022 Presented by Bruce Nagy, NAVAIR, NAWCWD China Lake.

inputs from the MLDL, e.g., memory-based learning (lazy learners) may need to carry the appropriate data set into the MLIL, and additional holdout data sets may be necessary to achieve MLIL verification robustness. That is, the MLDL may need to be modified for different types of data-driven ML techniques to account for their unique aspects; however, minimally equivalent processes, objectives, activities, and output data items should be accomplished, e.g., memory-based learning (lazy learners) may need to indicate the impact of frozen/non-adaptive and supervised assumptions.

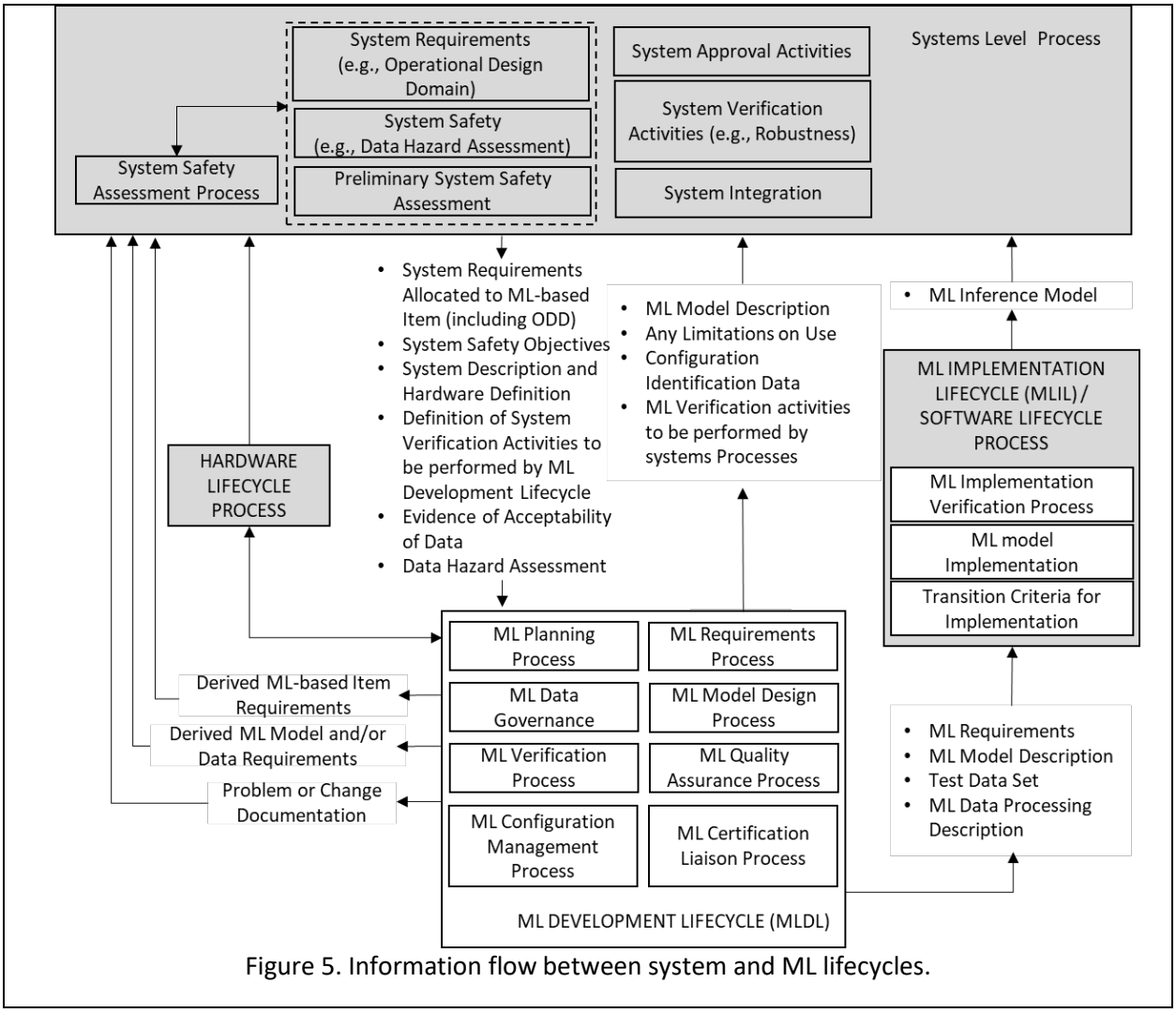

Figure 5. Information flow between system and ML lifecycles.

Notes for Figure 5:

- Note 1: Grey fill is used to illustrate existing traditional processes, i.e., processes covered under existing standards.
    - Systems Level Process, as indicated by AMACC Section 4, is covered under ARP 4754 and ARP 4761
    - MLIL/traditional software lifecycle process is covered under DO-178C.
    - Hardware implementation is covered under DO-254, with possible updates or supplements to accommodate ML-specific chips or GPUs (follow-on).

- Note 2: White fill is used to illustrate processes newly added to address the certification of ML.
- Note 3: Illustration highlights the addition of the ML Development Lifecycle (MLDL) and the ML Implementation Lifecycle (MLIL).
    - ML Implementation Lifecycle is executed within the Software Lifecycle Process.
- Note 4: Traditional hardware and software lifecycles were intentionally excluded from this figure, as the purpose of this illustration is to keep the focus on the newly added ML Development Lifecycle (MLDL) and the processes in the ML Implementation Lifecycle (MLIL). Those traditional lifecycles would be followed by any traditional elements that are supporting the ML model in the ML-based item.
- Note: 5: Software Lifecycle Process only shows those processes unique to the MLIL. Traditional Software Lifecycle Processes, like Software Planning Process, Software Coding Process, Software Quality Assurance, Software Configuration Management, and Software Verification Process, are still executed during the MLIL, but not shown.

Section 2.1 System Requirements Allocated to ML

An ML-based system will follow the traditional aircraft system decomposition process from concept to preliminary design to detailed design. That process will be augmented as necessary to account for the unique system considerations necessitated for ML systems, e.g., primary, secondary, and emergency signal/source information, failover planning for unacceptable uncertainty determinations, ML maintenance, mishap information for incident recreation, integrity voting schemes, mitigations (e.g., continuous monitoring, runtime assurance), operational design domain operations resiliency, and operational design domain updates (e.g., retraining). Those unique ML considerations should be included from concept development through to item development, and in all supporting system safety assessment process considerations.

Note that the system requirements are often decomposed into functional and non-functional requirements. Functional requirements refer to what the sub-system or item is expected to do and how well it is expected to perform. Examples are amount and type of data available, required accuracy of classifying objects or estimating a parameter, and sensitivity or robustness metrics. Non-functional requirements refer to the environment or constraints that the sub-system or item needs to be able to operate under. Examples of these constraints of the operating environment are the following: temperature, humidity, altitude, dust level, brightness or contrast level, etc. and available memory and throughput.

The system lifecycle process and the system safety process outlined in ARP 4754 and ARP 4761, respectively, are still applicable to the development of the ML software item. As shown in Figure 6, there is an interaction between the systems development processes and the system safety processes.

The system lifecycle process identifies and allocates system requirements to items based on the aircraft functions, system architecture, and architectures of subsystems. Those functions and architectures, specifically the systems architecture, determine which system requirements are allocated to software and hardware items. By establishing systems, associated interfaces are also identified between those systems. All inputs and outputs associated with those interfaces should also be identified. Those interfaces may serve to provide input data signals and sources, which contain necessary features and attributes, to the ML-based system/sub-system/item.

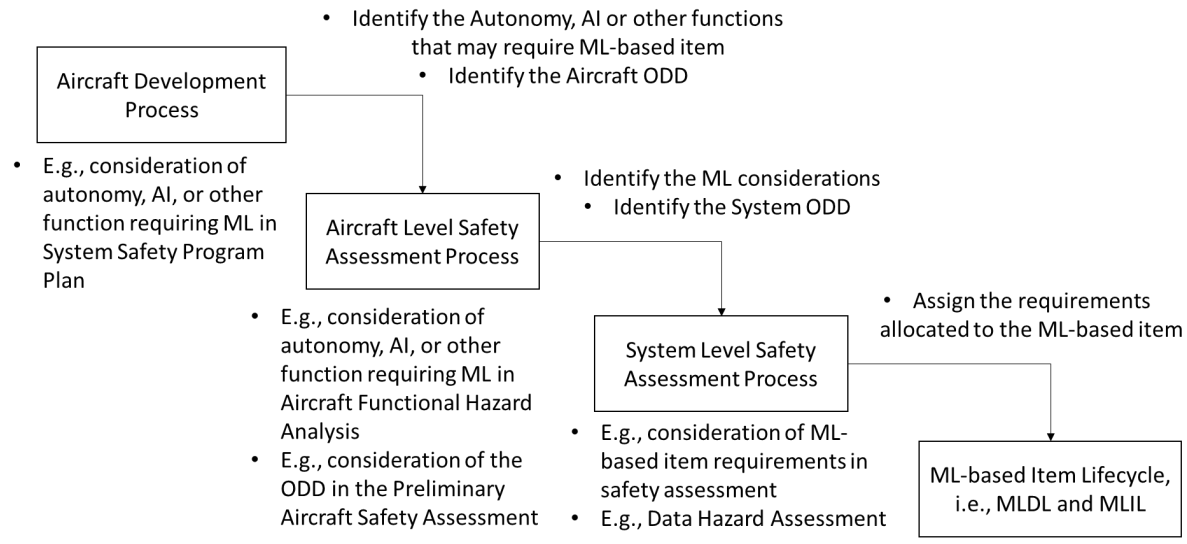

Figure 6. Safety assessment process interaction with the system development process.

The traditional ARP 4761 system safety assessment process consists of the following processes: Aircraft Functional Hazard Assessment (AFHA), Preliminary Aircraft Safety Assessment (PASA), System Functional Hazard Assessment (SFHA), Preliminary System Safety Assessment (PSSA), System Safety Assessment (SSA), and Aircraft Safety Assessment (ASA).

For the safety assessment process, the following are traditional safety assessment methods: Fault Tree Analysis (FTA), Dependence Diagram, Markov Analysis, Model Based Safety Analysis (MBSA), Failure Modes and Effect Analysis/Summary (FMEA/FMES), Cascading Effects Analysis, Zonal Safety Analysis, Particular Risks Analysis, and Common Mode Analysis.

For ML, new safety assessment processes may be needed, e.g., Bayesian Belief Networks used to augment Fault Trees, Event Sequence Diagram Analysis for assessment of scenarios, and using Quantitative Safety Metrics for ML performance and establish Safety Performance Metrics. One area where a new safety assessment must be accomplished is data, i.e., data hazard assessments must be conducted to determine the safety impact of the data. Moreover, some of the current safety assessment methods may need to be augmented for ML concerns, e.g., the inclusion of the ODD in hazard assessments.

ML functions and considerations should be included in the system safety assessment processes, so that all anticipated ML functionality is included through all the processes and methods listed above. As shown in Figure 7, by including the ML functional considerations through the aircraft system and system safety design decomposition processes and activities the systems and safety requirements will be allocated to the ML-based software item and the development assurance level of the ML functions will be determined.

Through the system design and system safety processes the failure conditions of the system, especially those due to the ML functionality, should be identified, assessed, and mitigated. In addition, the safety requirements should capture the mitigation requirements necessary to ensure that the integrity of

system safety is maintained as hazards from the ML functionality are encountered. Those system safety requirements are flowed to the items.

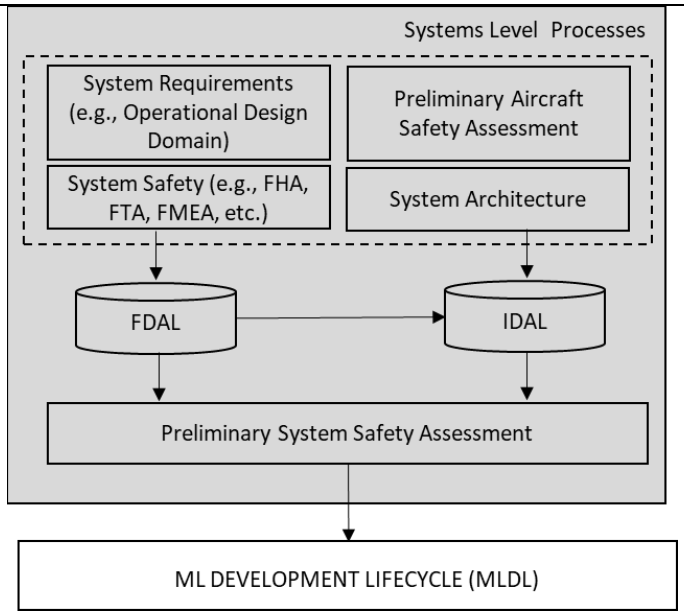

Figure 7. Approach for FDAL/IDAL assignment process.

System requirements allocated to the ML-based software item, including safety-related ML requirements, are developed, and refined into ML-based software item requirements that are verified by the ML implementation verification process, e.g., verification executed in the MLIL. The ML-based software item requirements and the associated verification should establish that the ML-based software item performs its intended functions under any foreseeable operating condition and operational design domain, including robustness conditions.

System requirements allocated to ML-based software items should include:

- System Requirements Allocated to ML-based software item, e.g., performance, interface, constraints, design methods, partitioning, dissimilarity, redundancy, monitoring,
- ML System Safety Objectives, e.g., certification requirements,
- System Description and Hardware Definition,
- Definition of System Verification Activities to be performed by ML Development Lifecycle,
- Evidence of Acceptability of Data Sets,
- Acceptable probabilistic behavior, e.g., acceptable residual uncertainties, confusion matrix values, and acceptable percentage of policy violations, etc.

Section 2.2 Information Flow Between System and ML Lifecycle Processes

As shown in Figure 5, there is information flow between the ML lifecycle processes and the systems lifecycle processes. Due to the interdependency of the ML-based software item on traditional software items, hardware items, systems and sub-systems, iterations should be expected as the systems

development process progresses. Adequate configuration management at the ML-based software item level and systems development process level should track all iterations. Assurance through the configuration management process should ensure alignment across all items and systems when iterations and updates occur.

Section 2.2.1 Information Flow from System Processes to ML Processes

The information flow from system processes to the ML-based software item process is similar to the information identified in DO-178C Section 2.2.1. However, the details are updated as follows:

a. The systems process should provide the system requirements allocated to ML-based software item.
b. The systems process should provide the requirements for the operational design description (ODD), edge cases, corner cases for the ML-based item.
c. The systems process should provide data hazard assessment to the ML-based item.
d. The systems process should provide details of continuous monitoring and mitigations of the ML-based item.
e. The systems process should provide details of system verification activities to be performed by ML Lifecycle.
f. The systems process should provide details of evidence expectations for the ML data assurance and ML model assurance.
g. The systems process should provide details traditionally passed to the item, e.g., system architecture description, item development assurance level, partitioning details, interface detail, target hardware, etc.

Section 2.2.2 Information Flow from ML Processes to System Processes

The information flow from ML lifecycle processes to system is similar to the approach discussed in DO-178C Section 2.2.2. For the purposes of the ML lifecycle, the following are highlighted as provided back to the systems processes:

a. The ML lifecycle process should provide back to the systems process any derived ML requirements.
b. The ML lifecycle process should provide back to the systems process derived ML data requirements and/or ML model requirements.
c. The ML lifecycle process should provide back to the systems process problem and/or change control documentation.
d. The ML lifecycle process should provide back to the systems process any limitations on use of the ML-based item.
e. The ML lifecycle process should provide back to the systems process program planning and accomplishment documentation, e.g., Plan for Machine Learning Aspects of Certification and the Machine Learning Accomplishment Summary.
f. The ML lifecycle process should provide back to the systems process the Machine Learning Configuration Index.
g. If necessary, the ML lifecycle process should provide back to the systems process the verification activities to be performed by systems processes.

Section 2.2.3 Information Flow between ML Processes and Hardware Processes
In general, the information flow between ML processes and hardware is similar to that detailed in DO-178C Section 2.2.3.

Unique information flow between the ML process and the hardware process should include any hardware or software considerations related to certification considerations for Multi-Core Processing[34] or ML specific processing considerations, e.g., use of general-purpose GPU and/or TPU considerations for executing ML inference models.

Section 2.3 Systems Safety Assessment Process and ML Level
The ML-based software item development assurance level (IDAL) should be determined through the system safety assessment process laid out in ARP 4754 and ARP 4761, appropriately augmented, when necessary, to include considerations for autonomous behaviors, i.e., classification, type, and level.

As laid out in ARP 4754 and ARP 4761, in general, the system function and system architecture determine the development assurance level. Architectural decisions can be made to reduce the DAL assigned to individual items, e.g., partitioning, redundancy, monitors, etc. Such decisions should be determined during the systems lifecycle process, and not during the ML lifecycle.

As with traditional software item development, using partitioning (as discussed in ARINC 653) for ML allows all items, e.g., traditional software items and ML-based software items, within a partition to be assigned the DAL of the highest item within the partition.

The IDAL assigned to the ML-based software item establishes the level of rigor necessary to demonstrate appropriate assurance to certifying authorities.

As with traditional software, the development of the ML-based software item to the appropriate development assurance level does not imply a certain failure rate to be associated with the ML-based software item. Given the uncertainty and probabilistic nature of ML[35,36,37], this disclaimer is even more important for ML. Development assurance levels processes, objectives, and activities help guarantee that the ML-based software item will perform as designed and minimize the unexpected behavior and epistemic uncertainty[38]. The system requirements allocated to the ML-based software item should indicate the probabilities and uncertainties, both acceptable and unacceptable, of ML-based software item performance. That is, the system requirements should indicate the performance assurance level, e.g., necessary accuracy, true positives, and true negatives rates for ML algorithm.

The ML lifecycle planning process should identify the assurance levels and the level of rigor for the ML-based software item. During the planning process the development assurance level should be brought

[34] For example, FAA AC/AMC 20-193, which is the new replacement for CAST-32A handling Multi-Core Processing development and certification, or ARMY SRD Multi-core Processing guidance.
[35] Ghahramani, Z. (2015) Probabilistic machine learning and artificial intelligence. Nature, 521:452–459.
[36] Jiang H., Kim B., Guan M., Gupta M., To trust or not to trust a classifier, Advances in Neural Information Processing Systems (2018), pp. 5541-5552.
[37] Malinin A., Uncertainty Estimation in Deep Learning with application to Spoken Language Assessment, (Ph.D. thesis), University of Cambridge (2019)
[38] Pereira A, Thomas C. Challenges of Machine Learning Applied to Safety-Critical Cyber-Physical Systems. Machine Learning and Knowledge Extraction. 2020; 2(4):579-602. [URL: https://doi.org/10.3390/make2040031].

to the attention of certifying authorities to ensure they agree. In addition, the ML-based software items' DAL is determined by and should align with the system safety assessment DAL determinations.

Should the output of the MLDL indicate that additional hardware or software items are necessary to enable the ML-based item to meet its functional requirements, then systems safety reassessment should be conducted to determine impact on the DAL and safety requirements.

Section 2.3.1 Relationship between ML Mistake and Failure Conditions

Failure and fault conditions occur with traditional software items and will also occur with ML-based software items. The root of failures and faults are mistakes that lead to errors within the item. When mistakes occur, they can lead to errors that ultimately result in faults and failures at the system and aircraft level. The purpose of development assurance is to establish multiple levels of processes, objectives, activities, and reviews to be able to identify these mistakes and remove them before they become errors. Example sources of mistakes for the ML lifecycle can be missing or incorrect ML-based item requirements, missing or incorrect ML data set development, incorrect ML model training, etc. This cascade of mistakes leading to errors and errors and then to faults and failures is illustrated in illustrated in Figure 8. When applied correctly and fully, the development assurance approach, i.e., the Swiss Cheese model, has worked well to remove mistakes from software items for 40+ years. Thus, the approach proposed in this document builds off this development assurance approach for ML-based items.

Mistake and error free item development begins with requirements. Requirements allocated to the item from the system level must be correct and complete. They must provide adequate detail of what is needed from the item to meet the system requirements and also satisfy the systems safety assessment process. For ML-based items, the system design and safety assessment process should take measures to include considerations for uncertainty generalization of ML model performance when exposed to deployed data sets or adverse/robustness data sets. To accomplish this robust systems level requirements and system safety requirements allocated to the ML-based item must be developed. The system safety process should identify the requirements for the ML generalization for edge cases, corner cases, and outside the operational design domain. These are necessary since the ML generalization uncertainty is much more pronounced at and outside the boundaries of the operational design domain. Lacking this guidance from the systems level, mistakes in ML requirements and derived requirements development process and in the production of the ML data set and ML model design and development could occur.

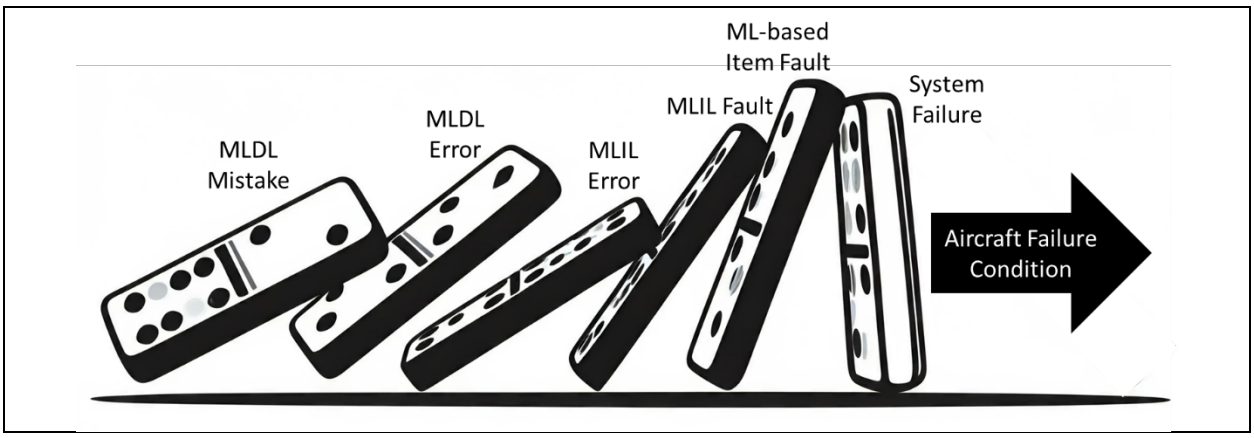

Figure 8. Sequence for a MLDL and MLIL Mistake Leading to Aircraft Failure Condition.

The proposed development assurance recommendations in this document are put forth to reduce the probability of mistakes occurring in the ML lifecycle. That is, the purpose of the MLDL development assurance proposed in this document is to help minimize and remove the mistakes through the ML development processes.

Section 2.3.2 Failure Condition Categorization

The failure condition categorizations used for traditional software item development are applicable for ML-based software item development. In addition, the potential impact is the effect of autonomy on the modulation of the development assurance categorization and ML-based item safety requirements. Another potential unique source for failure conditions is due to the deficiencies or erroneous content in the data set used for training, validation, and testing of the ML-based software item. A deficient or erroneous data set could introduce errors into the ML-based item which could reach deployment. Thus, data set assurance processes, activities, and objectives are introduced in this document, including data hazard assessment. The data hazard assessment should help identify the potential sources of hazards from the data set. For additional failure condition categorization, refer to appropriate system safety guidance standards, e.g., ARP 4761, AMACC, etc.

Section 2.3.3 ML Assurance Level Definition

As with traditional software item development lifecycle, the five levels of assurance levels are still applicable, i.e., Level A to Level E. Within this document the ML-based software item assurance level is equivalent to the IDAL definitions and levels. Those assurance levels used for traditional software item development, e.g., similar to those specified in MIL-STD-882E and DO-178C, have not changed for ML-based software item development, except for the definitions to be updated to account for ML:

- Level A: catastrophic outcomes with loss of life, permanent environmental impact, or more than $10 million in damages.
- Level B: critical outcomes with permanent injury, substantial environmental impact, or $1-10 million in damages.
- Level C: marginal outcomes with injury, environmental impact, or $1 million - $100K in damages.
- Level D: between marginal and negligible.
- Level E: negligible outcomes with minimal injury, minimal environmental impact, or less than $100K in damages.

The above definitions are only provided as guidance, and are addressed more completely in the appropriate system safety guidance standards, e.g., ARP 4761, AMACC, etc. Those foundational documents will take precedence in determining the formal definitions, while this document serves as guidance on these concepts.

Section 2.3.4 ML Assurance Level Determination

During the systems safety assessment process, an IDAL is assigned to the ML-based software item responsible for executing a number of functions. Those functions were assigned an FDAL during the safety assessment process. The IDAL is an aggregation of those functions and their associated FDAL. That is, traditionally the item is assigned the highest FDAL of the function set to be executed by the

item. This process should also be followed to determine the ML software item assurance level. Specifically, the IDALs are determined by the PSSA.

Section 2.3.5 ML Safety Requirements Determination

While the IDAL for an item is being established, safety requirements are also being determined. The IDAL and safety requirements are critical to ensuring the item is developed to the appropriate rigor to ensure safe deployment and mitigations of failures and errors when hazards occur. For ML-based systems this will be even more important given the fragile nature of the ML algorithms. The sensitivity of ML algorithms to bad, erroneous, and missing sensor, signal, feature, and attribute information must be considered during the systems safety assessment process. The system safety requirements should provide additional measures that should be taken to ensure robustness of the ML-based system due to the data sensitivity concerns as well as other weaknesses of the ML approach. To account for this concern the PSSA should generate ML-based item safety requirements, that include reliability, integrity, independence, performance, and separation requirements.

Section 2.4 Architectural Considerations

All traditional software item architectural considerations are still applicable, i.e., partitioning, serialization, independence, dissimilar software, etc. Additional architectural considerations applicable to ML-based software items are the following:

- Appropriately monitor and process input data.
    - Designing for data signals, sources, features, and attributes that may be at or outside the required operational design domain
    - Designing for missing data or data dropouts
    - Designing for data drift
    - Designing for robust data effects
- Appropriately monitor and process output data.
    - Designing continuous monitoring to account for potential uncertainty in the generalization of the ML algorithm
    - Designing run-time monitoring of the ML model in areas of adverse operational data, e.g., edge cases, corner cases, novelty data, and outlier data.

Other good ML-based item development practices are the following:

- Aim for the simplest design.
- Avoid unnecessary coupling dependencies.
- Remove time series dependencies.
- Delegate functionality to traditional systems.
- Avoid “end-to-end” architecture with direct control, i.e., avoid on-the-loop or in-the-loop but instead pursue modular pipeline architecture with advisory output.

Section 2.4.1 Input Data Monitoring

ML-based systems, subsystems, and items may benefit by instituting input data monitoring and adjudication protocols. Those protocols would take measures to ensure the ML-based software item is prevented from receiving data signals, sources, features, and attributes at and beyond the boundaries of

the operational design domain. Such measures would prevent the ML-based software items from generalizing on data outside the operational design domain, which is reflected in the data set used for ML model training, validation, and testing. In general, ML generalization typically performs poorly, and sometime unexpectedly, outside of the operational design domain. Taking measures to adjudicate such situations would help prevent unexpected and poor generalization behaviors.

The ML Data Processing Description will provide the design details of the expected input data for the ML-based item. Thus, any input data outside of the design details of the ML Data Processing Description could be considered outside the operational design domain.

Section 2.4.2 ML Safety Monitoring (e.g., continuous monitoring)

Like traditional software item safety monitoring, ML-based item safety monitoring can be equally beneficial. For an ML-based item, safety monitoring is a continuous monitoring of the output of the ML-based item. As discussed in DO-178C Section 2.4.3, safety monitoring checks for output software item (traditional or ML) conditions that would lead to failures in systems or subsystems.

The paper "Certification Considerations for Flight Critical Complex Non-traditionally Developed Software"[39] examines the use of safety monitoring techniques for ML. That paper stressed the importance of the following:

- Even with the application of safety monitoring, safety level of rigor and airworthiness development assurance processes should appropriately produce output data items for the non-traditionally developed software, e.g., ML (data-driven ML).
- Safety monitoring should be developed to the appropriate assurance level established by the system safety assessment process.
- Safety monitoring, i.e., continuous monitoring and mitigations (e.g., run-time assurance), could be applicable for instances where residual risk is unacceptable, after producing the assurance artifacts for non-traditionally developed software, e.g., ML (data-driven ML).
- The system safety assessment process should show that the safety monitor, i.e., continuous monitoring and mitigations (e.g., run-time assurance), is independent and not susceptible to the same common mode failure causing the ML-based software item to fail.

Safety monitoring techniques, i.e., continuous monitoring and continuous mitigations (e.g., run-time assurance), could be applied to mitigate situations where the input to or the output of the ML-based item is unacceptable. In the first case, the ML safety monitoring determines when the input data set is outside the ML-based item operational design domain. In such cases the ML safety monitor, i.e., continuous monitor, could restrict the input, so the ML-based item does not receive unacceptable input data. For the second case, another potential use of the safety monitor occurs when the ML output is outside of acceptable thresholds, in such a case the ML safety monitor, i.e., continuous mitigations (e.g., run-time assurance), could remove the ML-based item from the output and instead use a traditionally developed acceptable solution. In these cases, the continuous monitor could detect the violation and then the continuous mitigation could intervene to provide a mitigated solution. That is, the continuous monitor and mitigation is used to ensure the ML-based item does not introduce erroneous or

[39] "Certification Considerations for Flight Critical Complex Non-traditionally Developed Software", Accessible via dtic.mil (Accession Number: AD1160286). Publication Date: 2022-02-04. Also going through Army Public Release (ECD: 2022-05-04). Can be provided per request: jason.rupert@mtsi-va.com

unexpected outputs when encountering data sets outside the operational design domain or produce unacceptable output that may cause harm to the subsystem or system.

Section 2.5 ML Considerations in System Lifecycle Processes

The following section provides a listing of ML-based software items that should be considered by the systems lifecycle:

1. Parameter Data Items
2. User Modifiable software
3. Commercial-off-the-shelf ML models
4. Option-selectable software
5. Field-loadable software
6. ML Considerations in System Verification
7. ML Reuse – Models and Data

Section 2.5.1 Parameter Data Items

As described in DO-178C Section 2.5.1 for traditional software item, parameter data items are separate configuration items that influence the behavior of the code without modifying the Executable Object Code. Parameter data items are not part of the executable object code and are comprised of individual elements that consist of types, ranges, or enumerations of values. For the ML lifecycle, if hyperparameters or other ML inference model attributes are stored as parameter data items then the following considerations should be addressed:

- Controlling the modification of the parameter data item file during the MLDL and MLIL.
- Preventing the inappropriate modification of the parameter data item file during deployed operational use.
- Addressing enabling/disabling or activating/deactivating elements within the ML model from the parameter data item file.
- Establishing field-loadable guidance for the parameter data item.

The parameter data items should be assigned to the same DAL as the ML-based software item using them. That means the processes, objectives, and activities for the parameter data item should be appropriately carried out to ensure the validity of the parameter data item and that it receives the same level of rigor as that associated with the ML-based software item.

Section 2.5.2 User-Modifiable software

DO-178C Section 2.5.2 provides guidance for the traditional software item including user-modifiable features and functionality. All guidance provided by DO-178C Section 2.5.2 remains applicable for the ML-based software item. User-modifiable functionality should not be present unless the system requirements allocated to the software item (traditional or ML) identify the need for such user modification. Those requirements should specify the extent of modification allowed, and the system safety assessment should ensure that no user modification should cause harm to the system.

Section 2.5.3 Commercial-off-the-Shelf ML Models

DO-178C Section 2.5.3 provides guidance for the traditional software item using Commercial-off-the-Shelf (COTS) software. All guidance provided by DO-178C Section 2.5.3 remains applicable for the ML-based software item. That is, all guidance within this document is still applicable for COTS ML models,

e.g., full safety assessment and development assurance practices are expected to be followed and output data items made available to substantiate their completion.

Section 2.5.4 Option-selectable ML-based software item

DO-178C Section 2.5.4 provides guidance for the traditional software item using Option-selectable software. All guidance provided by DO-178C Section 2.5.4 remains applicable for the ML-based software item. Specifically, the systems development and systems safety analysis should ensure appropriate system, sub-system, and item requirements (traditional and ML, hardware, and software) are in place to support the safe execution of option-selectable software (traditional or ML). Moreover, protections should be in place in the ML-based software item to ensure inadvertent activation of erroneous configurations is not invoked when installed on the target computer.

Section 2.5.5 Field-loadable ML-based software item

DO-178C Section 2.5.5 provides guidance for the traditional software item using Field-loadable software. All guidance provided by DO-178C Section 2.5.5 remains applicable for the ML-based software item.

Section 2.5.6 ML Considerations in System Verification

System verification is covered in detail in ARP 4754 as item integration and verification. ML-based software items will not stand alone but instead will be enabled by traditional software and hardware items, as well as potentially other ML-based software items. Those items will be integrated and verified together as ML-based software items, ML-based subsystems, and ML-based system as needed to increase confidence and certainty in the overall ML-based aircraft verification process. Using an iterative integration and verification approach allows for feedback of performance and defects within the items, sub-systems, and system to be provided in a timely manner. In addition, different portions of the operational design domain can be presented to the ML-based item(s) at different iterations of integration.

If approved by the certifying authority, some item verification activities can be deferred to the systems verification activities, while some system verification activities, if approved by certification authorities, can be pushed down to item verification activities. If either of those approaches is pursued, they should be appropriately documented in the system and item verification plans, respectively.

Similarly, some hardware verification testing, e.g., multi-core processing and Real-time Operating System (RTOS) worst case execution time, may require coordination with the software item verification and hardware items verification. Such cases should be covered in the system verification planning activities.

Section 2.5.7 ML Reuse – Models and Data

ML reuse is briefly discussed in this section and is also addressed in Section 12.1 Use of Previously Developed Software.

FAA AC 20-148 and “Software Reuse in Safety-Critical Systems” [Leanna Rierson] are targeted at traditional software items, but much of the guidance remains applicable to ML-based software items. Primarily, those systems requirements designate that the software item (traditional or ML) is designed to be reused. Should the software item not be designed for reuse, chances are great that difficulty will be encountered when trying to certify the artifact for reuse in the new system.

For ML-based software items, some organizations[40,41] have indicated the difficulty and risk associated with attempting to reuse ML models. At a minimum retraining, retesting, and reverification will be needed should the data set change between reuse domains, but also the ML model algorithm may also be inappropriate or need modification as well. Not only retraining, but the cost of updating the interface code ("glue code") can be a hidden cost when attempting reuse.

For ML data sets, given the expense associated with the creation of a curated data set, reuse is encouraged. However, ML developers must be alert and not reuse data in such a way that data leakage[42] is allowed to occur. Data leakage can lead to misleading generalization results, especially thinking that performance is better than it is. Moreover, data sets can become stale and no longer applicable to the operational design domain and must be refreshed as needed.

Special considerations are necessary when pre-trained models are used during the training phase, where those considerations include, but are not limited to, re-validation against the new or updated data set and operational design domain.

Section 2.6 Systems Considerations in ML Lifecycle Processes

DO-178C Section 2.6 provides guidance for System Considerations in Software Lifecycle Processes for the traditional software item. All guidance provided by DO-178C Section 2.6 remains applicable for the ML-based software item.

Section 3 ML Lifecycle

As shown in Figure 5, the Machine Learning lifecycle consists of two lifecycles: (1) the ML development lifecycle (MLDL), and (2) the ML implementation lifecycle (MLIL).

Section 3.1 ML Lifecycle Definitions

Section 3.1.1 ML Development Lifecycle (MLDL) Definition

The ML development lifecycle is responsible for the development of the ML requirements, ML data processing document, and ML model description document through processing the data and a process of designing and training the ML model. Aligning the MLDL with DO-178C is accomplished through the following processes: planning process, environment set up process, requirements process, data governance process, model design process, verification process, configuration management process, quality assurance process, and certification liaison process. Section 4 details the objectives, activities, and outputs of these processes.

Section 3.1.2 ML Implementation Lifecycle (MLIL) Definition

The ML implementation lifecycle is responsible for the certifiable development and correctness of the ML inference model. This is accomplished by transitioning from the MLDL into the ML implementation

[40] Software Engineering for Machine Learning: A Case Study, Saleema Amershi, 2019, [URL: https://www.microsoft.com/en-us/research/uploads/prod/2019/03/amershi-icse-2019_Software_Engineering_for_Machine_Learning.pdf]

[41] "Hidden Technical Debt in Machine Learning Systems", D. Sculley, et al., Google, Inc., Advances in Neural Information Processing Systems 28 (NIPS 2015). [URL: https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf]

[42] Kaufman, Shachar et al. "Leakage in data mining: formulation, detection, and avoidance." KDD (2011), [URL: https://www.cs.umb.edu/~ding/history/470_670_fall_2011/papers/cs670_Tran_PreferredPaper_LeakingInDataMining.pdf].

lifecycle. Within the ML Implementation lifecycle (MLIL) will be a software item requirements process, software item coding process, integration process, verification process, and a continuation of the configuration management process, quality assurance process, and certification liaison process.

Those processes will follow the guidance provided for the traditional software item lifecycle, which is discussed in detail in DO-178C and DO-178C Supplements. Where necessary critical sections from DO-178C and Supplements are referenced within this document. Since references are made, the reader should also refer to the text of DO-178C to be aware of the in-situ context of the reference.

The MLIL could include the implementation of the ML model on ML unique hardware or GPUs, so RTCA DO-254 guidance as well as multi-core CPU and unique ML hardware guidance could also apply. As that is beyond the scope of this document, follow-on guidance will examine ML hardware concerns more thoroughly to determine appropriate ML hardware guidance.

If traditional software elements are used to support the ML-model in the ML-based item, then the development of those software elements will follow the full DO-178C lifecycle. Their development will merge with the ML-model development in the MLIL. All processes, plans, and procedures should account for this combination of traditional software development and ML-development process. Specifying the coordination of the traditional software element development process with the ML-development process is beyond the scope of this document. The remainder of this document focuses on the development of the ML-development process.

Section 3.2 ML Lifecycle Sequences

The MLDL and MLIL may or may not be accomplished by the same groups or companies. If they are accomplished by separate groups, this may require two separate sets of planning processes, configuration management processes, quality assurance processes, and certification liaison processes.

The sequence in which programs executed the various processes, and the objectives and activities associated with the processes will be unique to each program. Within a program's planning process, the sequence of processes, objectives, and activities should be established. That sequence should be provided to certification authorities for their concurrence.

For the ML lifecycle unique sequence considerations are associated with the model training activity within the ML Design Process and the data processing activity within the ML Data Governance Process. Given the possible uncertainty with having acquired an adequate amount of and appropriately representative data for training, iterations may be necessary between the data processing activity and the model training activity.

Figure 9 illustrates the example development sequences (and possible iterations) for the elements (ML-based and traditional) composing the ML-based software item. In comparison with traditional software items, Figure 9 illustrates the potential complexity involved, mainly due to the data collection and curation process, in the development of the ML-based software item.

In Figure 9 element A and B are ML-based software, while elements C and D are traditional software. Those elements are integrated together to form an ML-based software item. For each of the elements different development sequences were followed. For element A, the MLDL planning process was followed by the requirements process, data collection and curation, model training, model verification, MLIL planning process, requirements process, coding (against low-level requirements), verification, and

integration. For element B, the MLDL and MLIL planning processes were combined, an iterative approach to data curation and collection and to the model training process occurred. For element C, a traditional software development approach was pursued, i.e., planning, requirements process, design process, coding, verification, and integration. For element D, an iterative traditional software development approach was pursued before finalizing the requirements, design, coding, and integration. The diagonal arrows in Figure 9 represent combination of the elements into the ML-based software item, where each of the elements followed a unique lifecycle.

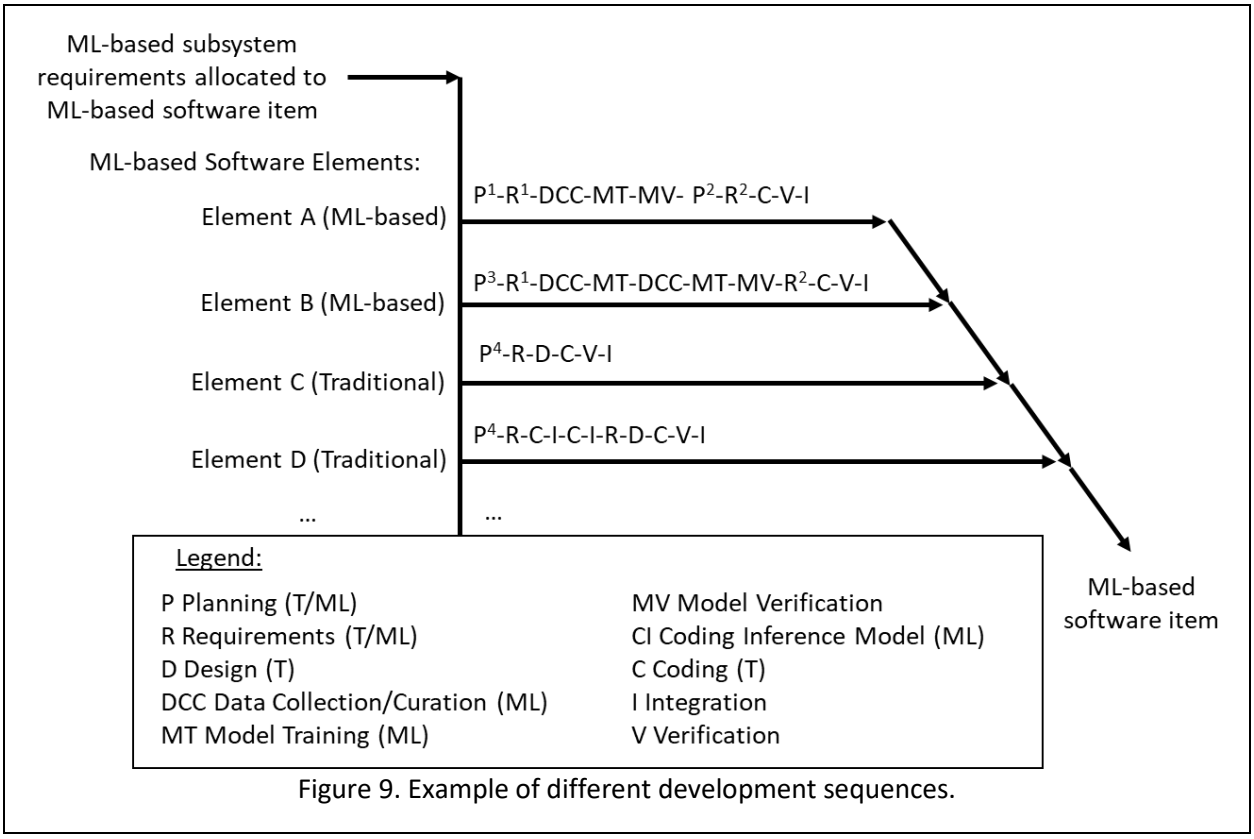

Figure 9. Example of different development sequences.

Notes for Figure 9:

- Note 1: P^1 indicates the MLDL planning phase.
- Note 2: P^2 indicates the MLIL planning phase.
- Note 3: P^3 indicates MLDL and MLIL planning phase were combined.
- Note 4: P^4 indicates the traditional software lifecycle planning phase.
- Note 5: R^1 indicates the MLDL requirements process.
- Note 6: R^2 indicates the MLIL/traditional requirements process.

Figure 9 is for illustration purposes only, and for simplicity intentionally left out many other processes, e.g., configuration management, quality assurance, integration testing, etc. This illustrates the flexibility that exists for processes sequences. That flexibility should be considered, and the approach appropriately captured in all planning materials.

Figure 9 also illustrates the possibility of multiple elements contributing to the construction of the ML-based software item. Those elements are a combination of at least one ML-based software element and

possibly one or more traditional based software elements. The ML-based elements are covered by the ML lifecycle which consists of the MLDL and MLIL described in this document, while the traditional software elements are covered by the DO-178C and supplements. The traditional software element may be responsible for data input pre-processing, data output post-processing, monitoring and mitigations, or other responsibilities.

Section 3.3 Transition Criteria

As indicated by DO-178C Section 3.3, for traditional software, transition criteria establish the threshold for transitioning from one process to another. For each process there exists input and output criteria. To transition into a process the input criteria for that process should be met. However, the input criteria need not be fully complete prior to transition. Similarly, to indicate a process as fully completed the output criteria should also be completed. However, transitioning into a subsequent process can begin once the transition criteria is met. Similar transition criteria exist for transitioning from the MLDL to the MLIL.

The planning process establishes the transition criteria and ensures that the airworthiness authority agrees with the transition threshold.

The traceability from the MLDL processes to the MLIL is critical so tight coordination is necessary between these two lifecycles to ensure proper transition from the MLDL to the MLIL. The transition criteria from the MLDL to the MLIL is the completion of the ML Requirements, ML Data Processing Description, and ML Model Description.

The planning process materials in the MLDL could cover the transition from the MLDL processes to the MLIL processes. The planning process materials should address that there are two separate levels of requirements and appropriate verification activities to occur, i.e., the MLDL requirements and verification, and the MLIL requirements and verification. In addition, the MLDL provides a lifecycle that is more flexible should iterative development be necessary prior to transitioning to a more rigorous MLIL which is less flexible to iteration.

Section 4 ML Lifecycle Planning Process

The ML lifecycle consists of the MLDL and the MLIL, where both lifecycles have planning processes. Those planning processes are similar but have some unique characteristics. Benefit is gained by addressing the MLDL and MLIL planning processes together in a single planning process activity. The following sections are written as if that approach is taken, so the term ML lifecycle planning process implies that both the MLDL and MLIL planning processes are considered.

The MLDL focuses on data collection and curation and model training to produce the ML Data Processing Description and ML model description document, while the MLIL focuses on coding the ML model description and testing the model for equivalent behavior to the MLDL trained model.

Greater consistency and traceability can be realized by accomplishing the MLIL planning process within the MLDL planning process, so that course of action is recommended. However, the MLIL planning process objectives, activities, and output data items can be addressed separately from the MLDL planning process objectives, activities, and output data items.

As shown in Figure 10, this section discusses the objectives and activities associated with the MLDL planning process. As with traditional software item lifecycle, this process produces the MLDL plans and standards that direct the MLDL development processes and the integral processes. Table A-1 of Annex A is a summary of the objectives and outputs of the MLDL planning process by software level. In Figure 10 cylinders are used to represent the output data items from the ML Planning Process. Those output data items serve as inputs for subsequent MLDL processes. In Figure 10, within the ML Planning Process, rectangles are used to show the objectives for the process.

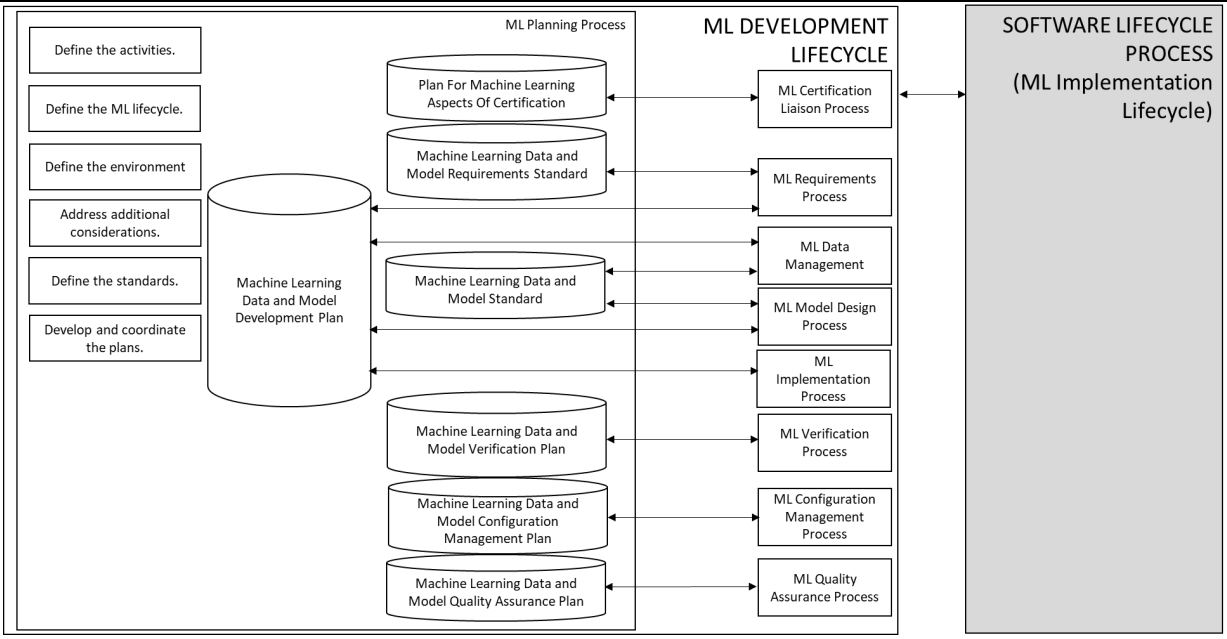

Figure 10. MLDL planning process.

Notes for Figure 10:

- Note 1: Figure 10 is only showing the MLDL Planning Process. MLIL Planning process is covered by DO-178C Section 4.0 and not repeated in this document.
- Note 2: Verification Process activities confirm conformance to the plans and standards.
- Note 3: Output data items, especially output verification data items, provide authentication artifacts to certification authorities.
- Note 4: ML lifecycle planning consists of the MLDL planning process and the MLIL planning process.

The ML lifecycle planning process covers the ML-based software item. As discussed previously, that ML-based software item consists of at least one ML-based software element and possibly one or more traditional based software elements. The planning associated with the ML-based elements are covered by the ML lifecycle planning process which consists of the MLDL and MLIL planning processes described in this document, while the traditional software element planning processes are covered by the DO-178C and supplements.

Section 4.1 ML Lifecycle Planning Process Objectives

The purpose of the ML lifecycle planning process is like the traditional software planning process listed in DO-178C Section 4.1, except the focus will primarily be on the MLDL planning processes. The MLIL planning processes can be handled jointly with the MLDL or separately. The following will uniquely identify the MLDL lifecycle planning objectives unique to the ML lifecycle:

a. The ML planning process, especially the MLDL planning process, should identify the processes, objectives, activities, and output data items of the MLDL process and the MLIL processes.
b. The ML planning process, especially the MLDL planning process, should identify the interaction and transitions between the ML data processes and the ML model processes, e.g., the transition from ML data set development to ML model training, validation, and testing, internal processes within the MLDL, and transition from the MLDL to the MLIL.
c. The ML planning process, especially the MLDL planning process, should identify the environment, languages, and tools used for the development of the ML data set and the ML model, and other tools necessary in the verification and configuration process.
d. The ML planning process, especially the MLDL planning process, should identify the novel items that could cause additional certifying authority consideration, e.g., usage and dependency on MCP CPU or GPU. (See Section 12 for further discussion)
e. The ML planning process, especially the MLDL planning process, should identify the ML data and ML model standards, which should be based on industry best practices for flight-critical data set and ML model development.
f. The ML planning process, especially the MLDL planning process, should identify the plans for ensuring the MLDL plans, processes, objectives, activities, and output data items comply with the guidance provided in this document.
g. The ML planning process, especially the MLDL planning process, should identify the plans for ensuring the MLDL plans, processes, objectives, activities, and output data items are under the appropriate configuration and revision control.

The appropriate planning process objectives for the MLIL are covered in DO-178C Section 4.1.

Section 4.2 ML Lifecycle Planning Process Activities

In general, the ML lifecycle planning process activities are similar to those discussed in DO-178C Section 4.2. The following activities are unique to the MLDL planning process.

Poor planning in the MLDL can lead to significant challenges further into the processes, objectives, and activities, so adequate effort should be put into up front planning to allow the MLDL to be a success. The activities for the MLDL include the following:

a. The ML lifecycle planning process activities, specifically the MLDL lifecycle planning process activities, should address the skillset, training, and experience of the staff supporting the ML data assurance and ML model assurance efforts during the MLDL for flight critical applications.
b. The ML lifecycle planning process activities, specifically the MLDL lifecycle planning process activities, should identify the schedule and programmatic plan for the MLDL processes, objectives, activities, milestones and transition criteria events.
c. The ML lifecycle planning process activities, specifically the MLDL lifecycle planning process activities, should identify the tool qualification and verification, validation, and accreditation

(VV&A) planned for the tools used for the ML data set development, ML model development, verification, and other processes, e.g., configuration management and quality assurance.

d. The ML lifecycle planning process activities, specifically the MLDL lifecycle planning process activities, should identify coordinating activities between the MLDL plans and the MLIL plans.
e. The ML lifecycle planning process activities, specifically the MLDL lifecycle planning process activities, should identify the configuration control, configuration management, and revision control approach for all processes, objectives, activities, and output data items in the MLDL.
f. The ML lifecycle planning process activities, specifically the MLDL lifecycle planning process activities, should identify any systems or ML based item architectural mitigation approaches, e.g., use of ML-based item continuous monitoring and mitigations.
g. The ML lifecycle planning process activities, specifically the MLDL lifecycle planning process activities, should identify the approach to address ML recommendations and concerns.
h. The ML lifecycle planning process activities, specifically the MLDL lifecycle planning process activities, should identify also address traditional software item concerns, e.g., parameter data items, and usage of deactivated code.

The appropriate planning process activities for the MLIL are covered in DO-178C Section 4.2.

Section 4.3 ML Lifecycle Plans

Within the ML lifecycle, the ML lifecycle planning artifacts consist of those associated with both the MLDL plans and MLIL plans. Where possible the output planning artifacts can be combined, e.g., the MLDL Aspects of Certification and the MLIL Aspects of Certification can be combined into a single ML Lifecycle Plan for the Aspects of Certification. Likewise, the MLDL and MLIL plans can be developed separately. Regardless, the ML lifecycle plans should appropriately detail the approach to be executed, i.e., combined or separately.

Section 4.3.1 MLDL Plans

The MLDL plans are similar to the approach established in DO-178C Section 4.3. Modifications of the plans are necessary to account for the unique development processes associated with data governance and data-driven ML model training. Where necessary those unique modifications are made to the MLDL plans.

The MLDL plans specify the organizations that develop the MLDL plans. The MLDL plans include the following:

- The Plan for ML Aspects of Certification (PMLAC) (see Section 11.1) defines the approach for the creation of the ML-based item, which includes indication of the approach for the ML data set and the ML model elements and supporting traditional software elements, and how recommendations are followed and concerns are mitigated.
- The Machine Learning Development Plan (MLDP) (see Section 11.2) defines ML data development and governance assurance process, ML model training and development process, ML data and model development environment, and how the ML development process objectives and activities will be satisfied.
- The ML Verification Plan (see Section 11.3) defines the verification of the ML data set and the ML model element of the ML-based item.

- The ML Configuration Management Plan (see Section 11.4) defines the configuration management of the ML data set and ML model, as well as the ML data set and ML model development environment and the repository of the configuration management systems.
- The ML Quality Assurance Plan (see Section 11.5) defines the auditing and sampling process to be followed to ensure the plans and standards are being appropriately followed to an appropriate level of completeness and representativeness, and to ensure the transition criteria are being fulfilled appropriately.

The MLIL PSAC, Development Plan, Verification Plan will cover the traditional software elements lifecycle and the integration of the traditional software elements with the ML-based software elements.

The MLDL plans should be provided to the airworthiness authority early in the process to ensure the approach agrees with all stakeholders.

The MLDL processes, objectives, and activities in the plans include the following:

a. The MLDL plans should include the processes, objectives, and activities, which are detailed in the subsequent chapters of this document and summarized in Annex A.
b. The MLDL plans should provide details on the inputs and outputs to the MLDL processes.
c. The MLDL plans should provide details on the tools used throughout the processes, and where tool qualification and verification, validation, and accreditation (VV&A) are necessary.
d. The MLDL plans should detail where and how deficiencies are captured and appropriately reported.
e. The MLDL plans should indicate the inclusion of configuration management and quality assurance in the processes.
f. The MLDL plans should indicate the process for proper interaction with the systems for completion of the ML lifecycle process, which includes the MLDL and the MLIL, and for problem reports to be relayed to the system process and systems safety process.

Section 4.3.2 MLIL Plans

MLIL Plans are based on DO-178C Section 4.3. Those plans should be appropriately tailored to account for the ML requirements, ML data requirements, and ML model requirements, along with the ML data description and ML Model Description from the MLDL to be used as inputs to the MLIL requirements process. Other appropriate tailoring should occur to the MLIL certification plan, development plan, verification plan, configuration management plan, and quality assurance plans. Those plans will guarantee the output MLIL inference model appropriately represents the required behavior, functionality, and performance captured in the ML requirements, ML data requirements, and ML model requirements.

Section 4.4 ML Lifecycle Environment Planning

The ML Lifecycle Environment planning is the process of identifying the MLDL environment and the MLIL environment. The planning of the MLDL environment and MLIL environment is recommended to occur in a combined planning approach, under the ML lifecycle environment planning approach. Having a combined MLDL environment and MLIL environment approach ensure transition from one environment to the other can occur without too many obstacles.

As with traditional software development, the development and implementation environment for ML can be a source of errors that contribute to failure conditions. Thus, the composition of the ML lifecycle environment may be influenced by the safety requirements determined by the system safety assessment, e.g., use of transitioning from interpreted programming language during MLDL to compiled programming language for MLIL.

As with traditional software development, the goal of the ML lifecycle environment selection will be the choice of programming languages, tools, methods, and hardware that minimize the possibility of failures being introduced. The goal of ML lifecycle environment selection is to choose mature environments that include appropriate safety features.

The MLDL environment is used for data governance processes and the ML model development. The MLDL environment consists of the set of programming languages, tools, methods, and hardware resources (including computer resources) used to specify, build, train, optimize, verify, describe, reproduce, and monitor the ML Model.

The MLIL environment planning process is captured in DO-178C Section 4.4. Given the MLIL is covered in DO-178C the focus of the following subsections will be on those environment planning activities unique to the MLDL environment planning process.

As with traditional software development, ML environment considerations that may affect the fault tolerant performance are the following:

a. The methods and notations used in the ML requirements process and ML design process.
b. The programming language(s) and methods used in the ML development, training, and coding processes, e.g., Python used for development and training during the MLDL, and C++ for the coding process during MLIL.
c. The ML development environment tools, e.g., PyTorch is used for development and training during the MLDL, and C++ custom library for the coding process during MLIL.
d. The ML verification and ML configuration management tools, e.g., in order to associate the version of the training data set, with all its features and attributes, to the version of the trained model a sophisticated data management version control tools may be necessary.
e. The need for tool qualification (see Section 12.2), e.g., tool qualification arguments for the MLIL will be more restrictive than those for the MLDL.

Section 4.4.1 MLDL Environment

The guidance provided by DO-178C Section 4.4.1 for traditional software development environment is also applicable to the MLDL environment. In addition, the selection of the MLDL environment should consider target deployment hardware environment requirements allocated to the ML-based item. Moreover, the environment should be selected for the development of the ML data set, e.g., synthetic data set or the simulation environment, based on the maturity of the tool and confidence it will not include any erroneous inputs. The same is true for any tools and programming languages selected for the development and verification of the ML model. The MLDL environment should be selected such that it will not insert error nor be absence of capabilities to cause errors through omission.

For example:

- An ML data set creation tool, e.g., COTS or open-source simulation environment, could introduce errors by inserting erroneous or fail to represent the ODD, which could insert an error in the ML model.
- An ML model framework tool, e.g., COTS or opensource model training framework, could introduce errors by producing corrupted parameters or fail to execute a layer and not reveal that error.

The MLDL activities for the selection of MLDL environment and tools should include the following:

a. The MLDL environment should include potential for the environment to input errors into the MLDL processes, especially the ML data or ML model development and verification processes.
b. The MLDL environment should include appropriate tool qualification and verification, validation, and accreditation of environments used for the creation of synthetic data sets and simulation environments.
c. The MLDL environment should include appropriate tool qualification of tools used during the ML model training process.
d. The MLDL environment should be supportive of the hardware deployment target required for the ML-based item from the systems process.
e. The MLDL environment used for verification of the ML data set and the ML model should be selected such that it will not erroneously overlook errors, and tool consideration is appropriately considered for tools used for MLDL verification.
f. The MLDL environment configuration for the environment and tools should be identified.
g. The MLDL environment and environment configuration should be under revision control.
h. The MLDL environment should include appropriate configuration management tools to place the MLDL plans and standards, ML requirements, ML data set and ML model, ML Data Processing Description, ML model description, and ML verification results under revision and configuration control.
i. The MLDL environment limitations should be identified and evaluated for adequacy in meeting the MLDL requirements.

In comparison with traditional software development, a unique MLDL tool qualification topic to address is the possible use of tools for creating the datasets used for training the ML model, and for building, training, and tuning the ML model. In such cases, special care should be taken to determine whether tool qualification is required, i.e., if the tool eliminates, automates, or removes processes, objectives, or activities of this criteria, and the output of the tool is not fully verified. If tool qualification is required, then the appropriate tool criteria should be assigned to the tool. Some considerations for assigning the tool criteria to such usages of tools used in MLDL are provided in "Considerations for Tool Qualification in Flight-Critical Applications using Machine Learning."[43].

Section 4.4.2 ML Lifecycle Programming Language Considerations

Different programming languages may be used for the ML model development during the MLDL and the MLIL. Those programming languages may differ in formality, i.e., where the programming language

[43] Carter, G., Vinegar, C., Terres, V., and Rupert, J., "Considerations for Tool Qualification in Flight-Critical Applications using Machine Learning", 43rd Digital Avionics Systems Conference (DASC), San Diego, California, USA. OCT 2024.

implementation and compilation occur. Interpreted programming languages, which may be used during MLDL, may be determined to be appropriate during MLDL, whereas compiled programming languages, which are typically used for high development assurance applications, may be determined to be necessary during the MLIL.

Interpreted programming languages do not compile the program down to machine executable code but instead are translated via an interpreter during execution into machine language. In most cases, execution of the interpreted source code requires the interpreted code and the language interpreter.

For compiled programming languages, prior to execution, the compiled programming language source code is compiled into machine language code via the language compiler, and then machine object files are linked into an executable via a linker, which is typical with the compiler. The machine executable code can then be executed independently of the compiler or linker.

In both cases supporting dependencies need to be met, i.e., appropriate library dependencies are included. Dependencies vary based on the environment, e.g., have dependencies on the operating system and hardware.

Historically interpreted programming languages have shown to have slower execution and less efficient than compiled programming languages. Interpreted languages have also had many qualities that have made them unable to meet the guidance provided in RTCA DO-332 for garbage collection, inheritance, and questions about the pedigree of their extensive frameworks. The choice of programming language is based on the amount of harm the programming language could introduce on the execution of the ML-based software item. The concerning characteristics of interpreted programming languages may prove to be an obstacle for use during the MLIL but may allow consideration during the MLDL. The reconciliation of these considerations and their impacts should be addressed in the planning process.

Section 4.4.2.1 MLDL Programming Language Considerations

The goal of the MLDL is the development of the ML model description. The selection of ML model programming language for the MLDL phase is important.Some interpreted programming languages include packages that optimize the performance and training of the ML Model. These features may not be directly traceable to the interpreted code but are inherent with the library or programming language. Those optimizations and considerations should be defined in the planning process to detect and verify this functionality. Moreover, these features and functionality should be appropriately documented in the ML model description.

The MLDL programming language selected should be robust, mature, stable, and have proper pedigree. Should those be absent then the programming language is invalid for use during the MLDL. For more information on MLDL programming language selection refer to Annex E-4.

If more than one programming language is used during the MLDL for the development of the ML model description, that should also be covered in the planning considerations.

Once MLDL programming language is judged to be adequate for the MLDL environment, it should only be considered acceptable for that product and not necessarily other ML environment configurations.

Section 4.4.2.2 MLIL Programming Language and Compiler Considerations
The guidance provided by DO-178C Section 4.4.2 for programming language and compiler considerations remains applicable for MLIL.

Section 4.4.3 ML Test Environment
ML test environment includes the MLDL test environment and the MLIL test environment. The MLDL test environment is expected to be more immature than the MLIL test environment, e.g., the MLDL test environment may involve general purpose computers, while the MLIL will involve target hardware. Both test environments should be adequately described, and the transition from the MLDL test environment to MLIL test environment to the target hardware should be described.

During ML test environment planning, considerations should be made for using the data sets. The data set should be divided into at least three independent data sets for training, validation, and testing. For some types of ML model development, the training and validation data sets will be used during the MLDL, and holdout data set will be kept for the MLIL, whereas for other types of ML model development data sets will be used differently. The ML test environments should be appropriately constructed to handle these various uses of data sets.

Due to the sensitivity of ML model performance at and beyond the operational design domain, the test environment should adequately represent that domain. The ML test environments should be representative of the operational design domain, which was identified in the system/subsystem requirements.

Section 4.4.3.1 MLDL Test Environment
Any differences between the MLDL test environment and the target computer should be identified. Differences considered should include throughput and memory estimates, which would ensure that the ML model can be executed in the allocated time. Any errors that could be introduced or failed to be detected because of these differences should also be identified when identifying the differences. To minimize these issues, the usage of a target computer hardware-in-the-loop testing environment is encouraged after MLDL design is completed.

Specific consideration should be given to any differences between the required operational design domain and the test environment. The effect of these differences on the ML model generalization should be determined and indicated. Appropriate verification plans should address these differences either through enhanced MLIL test environment, sub-system or system testing, or other augmented testing to fully parameterize the performance of the ML model when exposed to the operational design domain.

Section 4.4.3.2 MLIL Test Environment
The guidance provided by DO-178C Section 4.4.3 for test environment considerations remains applicable for MLIL.

Section 4.5 ML Development Standards
The ML development standards allow the ML-based software item vendor to indicate the rules and constraints they plan to follow over the MLDL and the MLIL. Standards in the MLDL are unique due to the ML data set assurance and model training assurance concerns, while the standards in the MLIL are those associated with the traditional software lifecycle.

Section 4.5.1 MLDL Development Standards

The MLDL standards include ML Data and Model Requirements Standard (ML D&M Requirements Standard) and ML Data and Model Standard (ML D&M Standard). The MLDL verification process uses these standards to ensure the output produced during the ML Data and Model Requirements Process and ML Data Governance and Model Training Processes are appropriate. Similar to DO-178C Section 4.5.1, the activities for the development of these standards include:

a. The standards comply with Section 11.
b. The MLDL standards enable the data and model components to be uniformly designed and implemented.
c. The MLDL standards provide further reinforcement that the ML-based software item is produced in accordance with safety requirements and in accordance with the verifiable methodologies.
d. The MLDL standards should include robustness considerations for the data sets and ML model.

Note 1: In developing MLDL data and model standards, consideration can be given to previous experience. Constraints and rules on data development, collection, processing, tagging, and dividing methods can be included to control data governance. In addition, model training practices may be considered to improve robustness and feature and attribute recognition generalization.

Section 4.5.2 MLIL Development Standards

The guidance provided by DO-178C Section 4.5 for standards remains applicable for MLIL.

Section 4.6 Review of the ML Planning Process

In general, the certifying authorities will want to review the planning process outputs, so thought should be given to the approach for those outputs to be reviewed. Traditionally, stages of involvement have been established to review the output data items at transitional process milestones in the program. Should the traditional stages of involvement be followed then the MLDL planning process and MLIL planning process artifacts would be reviewed together. This approach may not be advantageous to the program, so the planning aspects of certification should address the review strategy desired by the program.

Key activities that should be present regardless of the review approach desired:

a. Planning processes clearly identify how and when objectives and activities of this document are to be satisfied.
b. Planning processes materials are clearly and completely detailed such that they can be followed to produce consistent outputs.
c. Where independence is required, the planning process should identify how independence is being produced and maintained.
d. Methods are chosen so that a new process is not entered before the planning process details for that process have been reviewed by the certifying authority.

Section 4.6.1 Review of the MLDL Planning Process

MLDL planning process covers establishing the Data Governance Process and ML Model Training Process. The MLDL planning process activities include:

a. Data governance and ML model training processes occur consistently.

b. Data and ML model verification are shown to accomplish the objectives and activities provided in this document.

Section 4.6.2 Review of the MLIL Planning Process

The guidance provided by DO-178C Section 4.6 for planning process review remains applicable for MLIL.

Section 5 ML Lifecycle Requirements Process

As shown in Figure 11, the ML-based item lifecycle requirements are produced directly through analysis of system requirements allocated to the item, system safety requirements allocated to the item, and system/subsystem architecture. The ML lifecycle requirements process consists of the MLDL requirements process, which involves the development of MLDL data requirements and MLDL model requirements, ML Data Processing Description, and ML model description. The MLIL requirements process receives those MLDL outputs as inputs. The ML data description and ML model description are not shown in Figure 11, but instead is shown on Figure 13. Figure 11 does highlight that, in support of the development of the ML-based item, requirements may be allocated to traditional software items. Those requirements should follow the traditional software item lifecycle process as described in RTCA DO-178C.

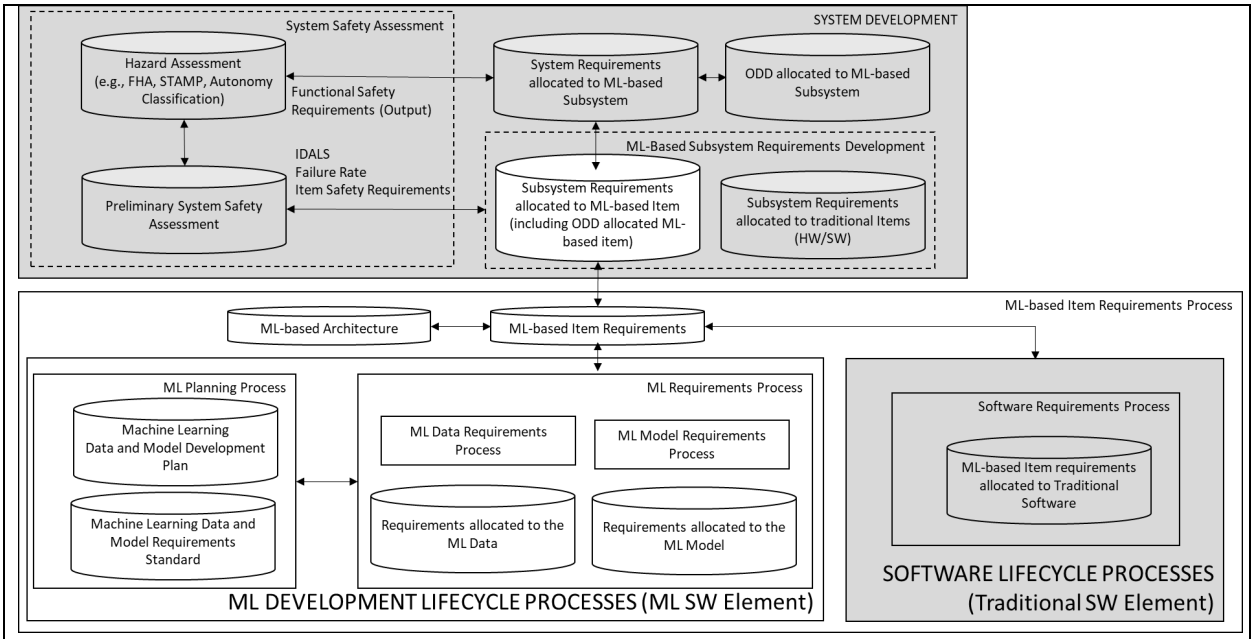

Figure 11. ML requirements process.

Notes for Figure 11:

- Note 1: ML Requirements Verification Process is listed in the MLDL Verification Process.
- Note 2: The ML-based software item may require traditional item(s) and elements (s).
- Note 3: Solid grey fill is used to indicate the traditional items.
- Note 4: Solid white fill is used to indicate the ML-based items.

The system and subsystem development process should produce the necessary pre-requisite inputs that are allocated to the ML-based item, and for the supporting traditional software and/or hardware items. The pre-requisite inputs from the system and subsystem development process allocated to the ML-based item should include the following:

- System requirements allocated to ML-based item, including ODD
- System safety objectives for the ML-based item, including the Data Hazard Analysis
- System Description and hardware definition
- Definition of system verification activities to be performed by ML Lifecycle, i.e., MLDL and MLIL

- Definition of acceptability of ML data coverage and ML model performance

Further definitions of these pre-requisites are covered in SAE ARP 4754 and SAE ARP 4761.

The Operational Design Domain (ODD), which is applicable to the ML-based subsystem and ML-based item, is defined during the system and subsystem development process. The applicable portion of the ODD is then allocated to the ML-based item through the requirements allocation process. This process ensures that the ODD applicable to the ML-based items is transferred from the systems and subsystems development processes to the ML-based item requirements process.

The ML lifecycle requirements may need to be allocated to traditional software elements, which are used to support the ML-based software item. Guidance for the development of the traditional software requirements is provided in DO-178C Section 5.1, whereas the requirements for the ML-based software item are provided below.

Further development of the ML requirements may be necessary before allocating them to elements, thus producing one or more successive levels of requirements. Specifically, the further development of ML lifecycle requirements may be required when the ML-based software item consists of ML-based software elements and traditional software elements.

The development (e.g., data processing, training, implementation, and verification) of the ML-based software item directly from the allocated system requirements, system safety requirements, and system architecture is strongly discouraged. Instead, further refinement of the allocated system requirements through the MLDL to MLIL is encouraged, i.e., production of ML model and data requirements. Further refinement is recommended due to the level of uncertainty, e.g., epistemic uncertainty, associated with the performance of the ML-based software item, i.e., ML-based software item due to its inherent probabilistic nature carries with it a level of uncertainty. The lack of adequate requirements decomposition only adds to the uncertainty and possibility of unexpected, undesired, or erroneous behavior.

Section 5.1 MLDL Requirements Process

MLDL requirements process consists of developing requirements and allocating requirements to the MLDL Data Requirements Process and the MLDL Model Requirements Process. Additional refinement of the requirements allocated to the ML-based software item may be necessary for that allocation to occur.

MLDL requirements are directly produced from those assigned to the ML-based software item during the ML lifecycle requirements process. Usually, those requirements will require further refinement prior to being allocated to the ML Data Requirements or ML Model requirements. Thus, one or more successive levels of requirements may occur before the allocation of ML data or ML model requirements.

Section 5.1.1 MLDL Data Requirements Process

MLDL data requirements process develops the requirements necessary for the data set used to train, verify, and test the ML model. The ML data requirements developed during this process should trace to the system requirements, system safety requirements, and system/subsystem architecture, especially the required operational design domain. These requirements should include considerations for precision, resolution, attributes/features, signal/source, robust/benign, and operational design domain.

Output from the data hazard assessment process should appropriately impact the development of the data requirements. The extent of the impact should be proportional to their severity. There should be bi-directional traceability from the data requirements to the data hazard assessment process, and vice versa.

ML data requirements process may also require feedback from the ML model development process, such as amount of data required and parameters or features needed in the data sets. Any such coordination should be identified in the planning process and appropriate plans.

Section 5.1.1.1 MLDL Data Requirements Process Objectives

MLDL data requirements have similar objectives to DO-178C Section 5.1.1.

The MLDL data requirement process objectives are the following:

a. The ML Data requirements, including derived ML data requirements, are developed and defined.
b. The derived ML Data requirements are provided to the system process for inclusion and evaluation by the system safety assessment process, especially the data hazard assessment process.

Section 5.1.1.2 MLDL Data Requirements Process Activities

The MLDL data requirement process activities are similar to the activities discussed in DO-178C Section 5.1.2. Additional activities are needed for MLDL Data to address the unique nature of ML Data during training, test, and verification.

The following are inputs to the MLDL data requirement process:

- System requirements, which include the operational design domain,
- System safety requirements, which include data hazard assessment,
- ML-based software item requirements,
- ML development plan, which includes the ML Data Development Plan, and
- ML Data Requirements Standard.

Once those are received the ML data requirements process can proceed.

The ML data requirements process should include the following:

a. The ML data requirements process should develop ML data set requirements that satisfy the systems requirements that detail the operational design domain (ODD), which includes nominal cases, edge cases, corner cases, adverse cases, and typical and robustness scenarios.
b. The ML data requirements process should satisfy the approach details for the development of ML data set requirements in the ML data development plan and be developed in accordance with the ML data requirements standard.
c. The ML data requirements process should ensure the ML data requirements are complete and representative of the operational design domain.
d. The ML data requirements process should include appropriate detail for the development of the ML data set, which may be a discrete data set or a simulation environment.

e. The ML data requirements process should include appropriate details to enable the ML data set to include the appropriate signals, sources, features, and attributes.
f. The ML data requirements process should appropriately address the ML concerns, e.g., simulation-to-real (sim-to-real) gap, and the ML recommendations.
g. The ML data requirements process should ensure the data requirements define the format, precision/accuracy, source/signal, and resolution of the ML data set.
h. The ML data requirements process should ensure data requirements are appropriate for the ML model algorithm selected.
i. The ML data requirements should be traceable to the systems requirements allocated to the ML-based requirements and the safety assessment, especially the data hazard assessment.
j. The ML data requirements process should provide derived ML data requirements to the systems process for adjudication.
k. The ML data requirements process should be under the appropriate configuration management and revision control.

The ML Data Requirements Data (see Section 11.9.1) is the output of the ML data requirements process.

Section 5.1.2 MLDL Model Requirements Process

The MLDL model requirement process defines the requirements for the ML model. The ML model requirements developed during this process should trace to the system requirements, system safety requirement, and system/subsystem architecture. They include the behavior expected within, at the boundary of, and outside the operational design domain, interfaces, and how model uncertainty will be presented.

Section 5.1.2.1 MLDL Model Requirements Process Objectives

MLDL model requirements objectives are similar to the objectives from DO-178C Section 5.1.1.

The MLDL model requirement process objectives are the following:

a. The ML model requirements, including derived ML model requirements, are developed and defined.
b. The derived ML model requirements are provided to the system process for inclusion and evaluation by the system safety assessment process, especially the data hazard assessment process.

Section 5.1.2.2 MLDL Model Requirements Process Activities

The MLDL model requirement process activities are similar to the activities discussed in DO-178C Section 5.1.2. Additional activities are needed for MLDL model to address the unique nature of the ML model during training, validation, and test.

The following are inputs to the MLDL model requirement process:

- System requirements, which include the ML model performance,
- System safety requirements, which include appropriate monitoring and mitigations,

- ML-based software item requirements,
- ML development plan, which includes the ML Model Development Plan, and
- ML Model Requirements Standard.

Once those are received the ML model requirements process can proceed.

The ML model requirements process should include the following:

a. The ML model requirements process should develop ML model requirements that satisfy the systems requirements that detail the ML model performance, which includes nominal cases, edge cases, corner cases, adverse cases, and typical and robustness scenarios.
b. The ML model requirements process should satisfy the approach details for the development of ML model requirements in the ML model development plan and be developed in accordance with the ML model requirements standard.
c. The ML model requirements process should ensure the ML model requirements are complete and representative for the operational design domain.
d. The ML model requirements process should include appropriate detail for the development of the ML model.
e. The ML model requirements process should appropriately address the ML concerns, e.g., interruptibility, and the ML recommendations, e.g., frozen.
f. The ML model requirements process should ensure proper performance representativeness of the ML model in the operational design domain is covered.
g. The ML model requirements process should ensure the data requirements account for robust and benign operational design domain considerations.
h. The ML model requirements process should ensure the ML model algorithm requirements are appropriate for the ML data set selected.
i. The ML model requirements should be traceable to the systems requirements allocated to the ML-based requirements and the safety assessment, especially the ML safety assessment considerations.
j. The ML model requirements process should provide derived ML model requirements to the systems process for adjudication.
k. The ML model requirements process should be under the appropriate configuration management and revision control.

The ML model Requirements Data is the output of the ML model requirements process.

Section 5.2 MLIL Requirements Process

In general, the MLIL requirements process aligns with the traditional software requirements process specified in DO-178C Section 5.1. The output of the MLDL is the ML requirements, ML model description, and the ML Data Processing Description. The ML requirements serve as the input requirements for the MLIL. The MLIL should use the ML requirements, ML Model Description, and ML Data Processing Descriptions as the inputs to begin the execution of the MLIL requirements process.

Section 6 MLDL Development Process

This section discusses the objectives and activities associated with the MLDL Development Process. These processes are applied as described in the MLDL planning process and the MLDL Development Plan. The MLDL Development process consists of the following processes:

1. ML Data Governance Process
2. ML Model Design Process
3. ML Verification Process

The MLDL Data Governance process, along with the ML model design process, are two critical new additions to the traditional software development processes to address the development techniques used to create ML models. These activities demand the application of critical thinking by those applying these techniques, so they are not falsely misled into believing better performance has been proven than is actually present[44]. Only through adequate objectives and activities being accomplished can data governance and ML model design be shown to provide adequate assurance to have confidence placed in their outcomes.

Likewise, the ML verification process, which includes data and model verification processes, will need to be adequately and robustly executed against the required operational design domain in nominal and robust conditions. Only through rigorous verification examination of the data set and the ML model can confidence be established that adequate training occurred for the model to correctly generalize on the entire operational design domain, especially during adverse and failure cases.

Section 6.1 MLDL Data Governance Process

As shown in Figure 12, the data governance process manages data sourcing, collection, processing, hazard assessment, and allocation. Data configuration management is addressed by the Configuration Management Process and data verification is addressed by the ML Verification Process.

There are numerous different and conflicting definitions of data governance and data management. The use of data governance in this document most closely aligns with that from Google, the medical field, and other indicated citations.

- Google indicates: “Data governance is a principled approach to managing data during its life cycle, from acquisition to use to disposal.”[45]
- Fothergill et al.[46], from the medical community, define data governance as the ”overall management of the availability, usability, integrity, quality, and security of data in order to ensure that the potential of the data is maximized while regulatory and ethical compliance is achieved within a specific organizational context.“ Eke, et al.[47], also from the medical

[44] That belief might be compared with the Dunning–Kruger effect.
[45] “What is Data Governance?”, [URL: https://cloud.google.com/learn/what-is-data-governance, Accessed: 10/12/2022]
[46] Fothergill BT, Knight W, Stahl BC, Ulnicane I. Responsible Data Governance of Neuroscience Big Data. Front Neuroinform. 2019 Apr 24; 13:28. doi: 10.3389/fninf.2019.00028. PMID: 31110477; PMCID: PMC6499198.
[47] Eke, D., Bernard, A., Bjaalie, J., Chavarriaga, R., Hanakawa, T., Hannan, A., Hill, S., Martone, M., McMahon, A., Ruebel, O., Crook, S., Thiels, E., Pestilli, F., “International data governance for neuroscience”, Neuron, Volume 110, Issue 4, 2022, Pages 600-612, ISSN 0896-6273, [URL: https://doi.org/10.1016/j.neuron.2021.11.017, Accessed: 10/12/2022].

community, expand upon this definition by indicating: "data governance as the principles, procedures, frameworks, and policies that ensure acceptable and responsible processing of data at each stage of the data life cycle, from collection, storage, processing, curation, sharing, and use to deletion. These procedures help maintain data integrity, quality, availability, accessibility, usability, and security and define data controllership (or stewardship) and other responsibilities related to the data."

- Janssen, et al. [48] provide guidance for the execution of a data governance program similar to that envisioned by this document: "Organizations and their personnel defining, applying and monitoring the patterns of rules and authorities for directing the proper functioning of, and ensuring the accountability for, the entire life-cycle of data and algorithms within and across organizations. … This definition takes into account both data and data processing by AI and other algorithms, considers that both data and algorithms change during their respective life-cycles, accounts for the personnel responsible for creating and use of data and algorithms, and adopts a systems (multi-organizational) view."

Data is one of the bigger technical debts[49] of the ML processing: "Data Dependencies Cost More than Code Dependencies" and "Changing Anything Changes Everything". These technical debts, i.e., resources and risks, are spread across all the processes associated with data governance, i.e., data planning to allocation. Data collection alone spans various types of data acquisition which could involve discovering existing collected data sets or synthetic generation and augmentation of data sets. After the collection process, the preparation and processing begin where labeling and other improvements are necessary. The labeling can be an intensive and technical process involving manual labeling or a semi-supervised labeling technique. Where necessary, improvements to the data may be necessary, which can also be intensive. Through each of these processes, care must be taken to maintain the data sets' validity and authenticity. The data set has a large impact on the performance of the model properly reflecting the required generalization behavior in the operational design domain. Benign and even imperceptible modifications to data can cause unexpected, unanticipated, and undesired behavior of the models when exposed to deployed native data sources.

The purpose of the data governance process is to ensure the data sets, e.g., training, validation, and testing, maintain authentic representative and completeness of the inputs required during the deployed operational design domain. The following objectives and activities ensure that authentic representative and completeness will be maintained through the data governance process.

[48] Janssen, M., Brous, P., Estevez, E., Barbosa, L., Janowski, T., "Data governance: Organizing data for trustworthy Artificial Intelligence", Government Information Quarterly, Volume 37, Issue 3, 2020, 101493, ISSN 0740-624X, [URL: https://doi.org/10.1016/j.giq.2020.101493, Accessed: 10/12/2022]

[49] "Hidden Technical Debt in Machine Learning Systems", D. Sculley, et al., Google, Inc., Advances in Neural Information Processing Systems 28 (NIPS 2015). [URL: https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf]

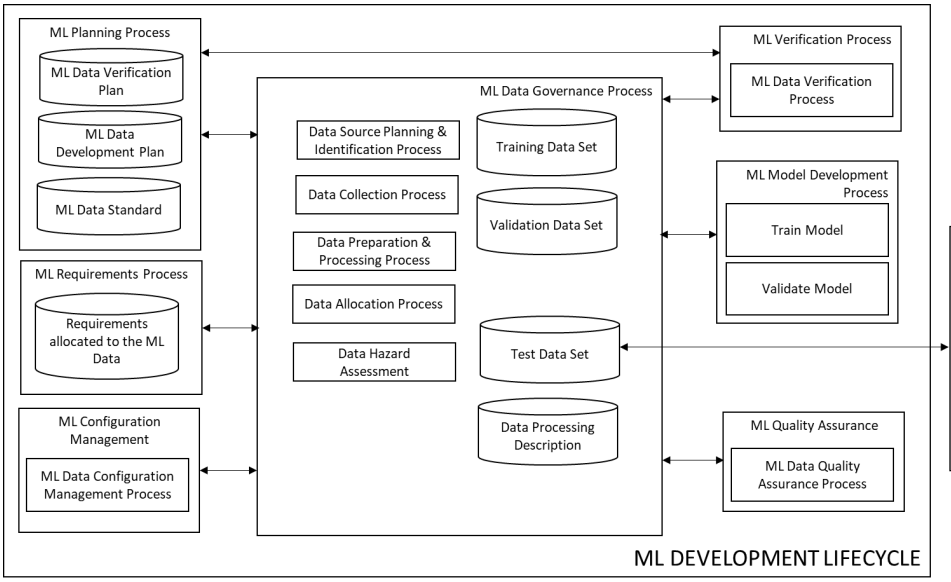

Figure 12. ML data governance process.

Notes for Figure 12:

- Note 1: Data set includes the features, attributes, and classes as well as the samples, signals, sources, and collection of those features, attributes, and classes.
- Note 2: Data configuration management, executed in the ML Configuration Management process, will ensure data is only used appropriately, e.g., avoiding data leakage and corruption, via methods like blockchain, and data integrity.
- Note 3: Data verification, executed in the ML Verification Process, will ensure the data is adequate, appropriate, representative, and complete as described in the Data Requirements.

Updates to the data set output may be driven by the ML Model Development Process or ML Verification Process. Should those updates occur the ML Data Requirements and system safety and system operational design domain requirements should be re-examined to determine if updates are necessary to those as well. Because requirements-based testing should be used for flight and safety-critical applications, any updates to those requirements should also be reflected in updates to verification cases.

The test data set is used within the ML Verification process, which includes the ML Data Verification and ML Model Verification processes, and in the MLIL, specifically the ML Implementation Verification process. Because data set changes can be costly, all efforts should be made to correctly produce fully representative and complete data sets.

Section 6.1.1 Data Governance Objectives

The objectives of the data governance process, which is part of the MLDL, are:

a. Data sets are developed in accordance with ML data requirements and standards.
b. At least three independent data sets are developed, e.g., training, validation, and test data sets are developed.

c. The ML Data Processing Description is developed and provided to the verification and implementation processes.

Section 6.1.2 Data Governance Activities

Inputs to the data governance process include the ML-based item requirements, system requirements, system safety requirements, the hardware interface and system architecture operational design domain from the system lifecycle processes, and the Machine Learning Data and Model Development Plan, the Machine Learning Data Requirements Standard, and the Machine Learning Data Requirements from the ML Data Requirements process.

The primary output of this process is the ML Data Processing Description (see Section 11.10).

The data governance process is complete when its objectives and the objectives of the ML model verification processes associated with it are satisfied. Activities for this process include:

a. The data governance process should be executed such that ML data requirements are completely and adequately met.
b. The data governance process should conform to the ML Data Standard.
c. The data sets should conform to the ML Data Development Plan and be verifiable, representative, complete, and independent.
d. The data source planning and identification process should identify the origin of the data sources.
e. Inputs to the data governance process detected as inadequate or incorrect should be reported as feedback to the input source processes for clarification or correction.
f. The data governance process should ensure system safety requirements, including data hazard assessment, allocated to the ML-based software item to preclude system hazards are adequately addressed and defined.
g. The data sets should be correct quantitatively in terms of attributes, features, signals, sources, tolerances, format, precision/accuracy, and resolution conformant to the MLDL data requirements.
h. The data sets should be complete, representative, and robust for the full operational design domain, e.g., nominal and off-nominal, i.e., edge case, corner cases, dropouts, missing, and corrupted attributes, features, signals, and sources.
i. The data governance process activities could introduce possible errors into the ML model. The use of data assurance, data hazard assessment, or other means may be necessary to augment the data governance process to ensure no erroneous or unexpected behavior will be injected into the ML model. In such cases, additional data processes should be documented in the ML Data Development Plan. For example, tool qualification of the data set creation tool and/or verification, validation, and accreditation (VV&A) of the data set simulation may be used. These measures help to further ensure that no erroneous or unexpected features, attributes, sources, signals, or entities are included in or missing from the data set. This precaution helps to prevent any negative impact on the ML model during the training, validation, or testing process by minimizing the possibility of inclusion of erroneous or unexpected behaviors or functions.
j. The data governance process should be executed in conjunction with the MLDL configuration management process to ensure data integrity.

k. The data governance process should be executed in conjunction with the MLDL quality assurance process to ensure all appropriate processes and procedures are adequately followed as identified in the ML Data Development Plan.
l. The data governance process should result in appropriate independent data sets, e.g., training, validation, and test data sets. In this case independence implies holdout (unseen) data sets, so the model has not seen the data before validation i.e., hyperparameter and parameter tuning, or testing is being performed.
m. The ML Data Processing Description output data item should adequately capture the data processing details.
n. If necessary (as required), reproduction of all data sets can be confirmed through independent verification and validation replication using the ML Data Processing Description.
o. The data governance process should indicate the approach to addressing the ML concerns, e.g., sim-to-real, and abide by the ML recommendations.

Section 6.2 MLDL ML Model Development Process

The ML Model Development process involves the design and coding of the ML model algorithm and then the training of that model algorithm on the training, validation, and test data set. The ML model development process should be based on the ML Model Development Plan, ML Model Standard, and ML Model requirements. The main outcome of the ML Model development process is the ML Model and the ML Model Description. The ML model description should include the ML model design and architecture details.

The ML Model Development Process may drive modifications and updates to the ML model requirements, ML model design, ML data sets (training and validation). Those artifacts should be updated appropriately, where the final reconciled ML model design is reflected in the ML model description.

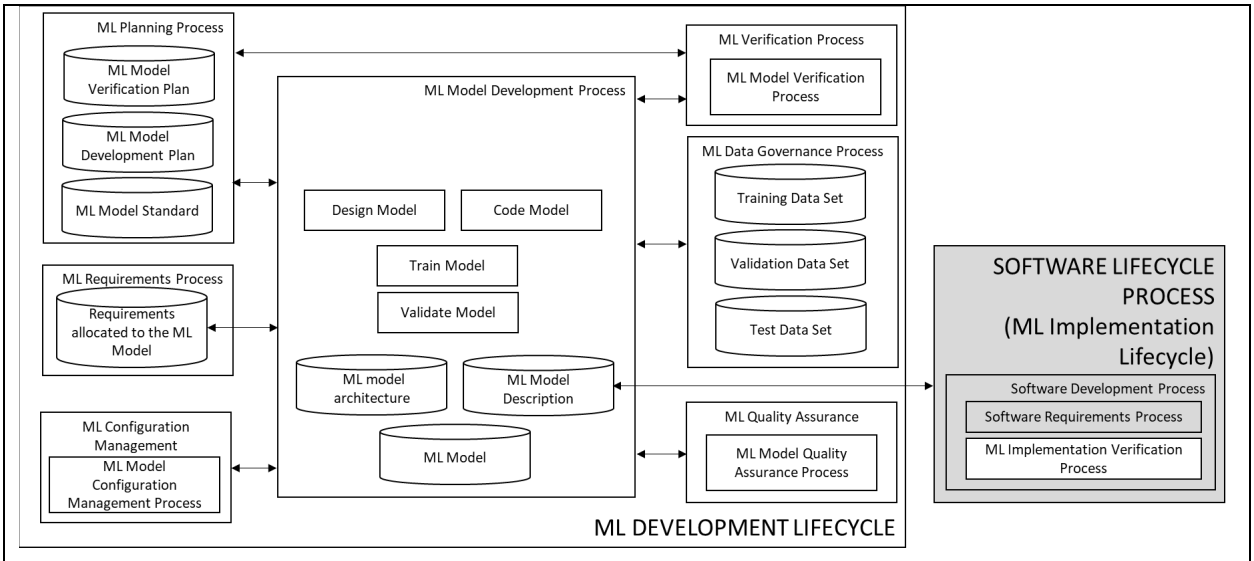

Figure 13. ML model development process.

Section 6.2.1 MLDL Model Development Objectives
The objectives of the model development process are:

a. Develop ML model architecture, i.e., design model activation functions, hyperparameters.

b. Develop the ML model, i.e., in accordance with ML model requirements, ML development plan and ML model standards.

b. Train ML model, e.g., train the model on the ML training data set.
c. Identify the ML model parameters and validate the ML model, e.g., using the ML validation data set determine the ML model parameters.
d. Develop the ML model description.

Section 6.2.2 MLDL ML Model Development Activities
Inputs to the model development process include the ML-based item requirements, system requirements, system safety requirements, the hardware interface and system architecture operational design domain from the system lifecycle processes, and the ML Data and Model Development Plan, the ML Model Standard, and the ML Model Requirements from the ML Model Requirements process.

The primary output of this process is the ML Model Description (see Section 11.11).

The MLDL model development process is complete when its objectives and the objectives of the ML model verification processes associated with it are satisfied. Activities for this process include:

a. The MLDL model development process should be executed such that ML model requirements are completely and adequately met.
b. The MLDL model development process, e.g., architecture, coding, training, and validation, should conform to the ML Model Development Plan and be verifiable, representative, and complete.
c. Inputs to the MLDL model development process detected as inadequate or incorrect should be reported as feedback to the input source processes for clarification or correction.
d. MLDL ML model architecture should ensure system safety requirements allocated to the ML-based software item to preclude system hazards are adequately addressed and defined.
e. The MLDL ML model architecture should be architected correctly in terms of being able to receive and process attributes, features, signals, sources, tolerances and type sample conformant to the MLDL model requirements and the training data set.
f. The MLDL model training should ensure the ML model is robust to missing and erroneous signals, sources, features, and attributes shown through evaluation against the validation data set.
g. The MLDL model training process activities could introduce possible errors into the ML model. The use of ML model learning assurance or other means may be necessary to augment the MLDL model development process to ensure no erroneous or unexpected behavior will be injected into the ML model. In such cases, additional ML model development processes should be documented in the ML Model Development Plan.
h. The MLDL model development process, if necessary, should allow for ML model validation, such that the internal framework hyperparameters of the ML model can be optimized.

i. The MLDL model development process should be executed in conjunction with the MLDL configuration management process to ensure model integrity and reproducibility, and to document design choices.
j. The MLDL model development process should be executed in conjunction with the MLDL quality assurance process to ensure all appropriate processes and procedures are adequately followed as identified in the ML Model Development Plan.
k. The ML model description output data item of the ML model development process should adequately capture the model design and development to the extent that it can serve as input to the MLIL requirements development process.
l. The ML model description should enable semantic reproduction of the model parameters, e.g., neural network weights, and hyperparameters, e.g., number of layers, and neurons in each layer.
m. The ML model description should enable ML model performance to be confirmed through verification and validation.
n. The ML model description should enable the independent production of the ML model in the MLIL. The MLIL will use the ML model description to confirm the ML inference model via verification and validation.
o. The MLDL model development process should address the ML concernsand abide by the ML recommendations, e.g., using a modular pipeline instead of an end-to-end pipeline.

To minimize the complexity of the ML model, an approach that uses smaller models, or even tabular solutions, that embrace a modular pipeline are recommended over the use of larger models in an end-to-end pipeline. This approach has qualitative benefits in the area of explainability and verifying alignment with the system safety objectives. With increasing model complexity also comes increased difficulty in verifying the model satisfies the system safety requirements.

Section 6.3 MLDL Verification Process

MLDL Verification is a twofold process: (1) determining that the software fulfills the requirements and (2) determining the software does not contain any unintended/unexpected functionality. The former focus is similar to traditional software verification, e.g., fulfillment of allocated systems requirements (functional and non-functional). For the latter, detecting erroneous behavior in ML is different from detecting those in traditional software, which necessitates novel test generation approaches. Further investigation of techniques beyond those listed below are necessary to identify verification approaches for ML that ensure no unintended behaviors are present. The following listings of MLDL verification processes provide an initial listing of approaches that should be considered. Evolution and refinement of these approaches are expected and anticipated as the community provides feedback on techniques proven to work more successfully and repeatably than others.

The MLDL verification is similar to the verification process described in DO-178C Section 6.0.

The MLDL verification process is completed by fulfilling the verification objectives, which are accomplished through verification activities. These MLDL verification objectives are an evaluation of the MLDL requirements process, the MLDL model architecture, ML data governance process, the ML model

development process, and finally the MLDL verification process. The verification process may involve testing, reviews, analysis, inspection, or other methodologies or combinations of verification techniques appropriate for the task.

As shown in Figure 14, MLDL Reviews and Analyses process includes the following verification activities:

- Verification, reviews, and analysis of MLDL Requirements
- Verification, reviews, and analysis of MLDL Architecture
- Verification, reviews, and analysis of MLDL Data Sets
- Verification, reviews, and analysis of ML Model

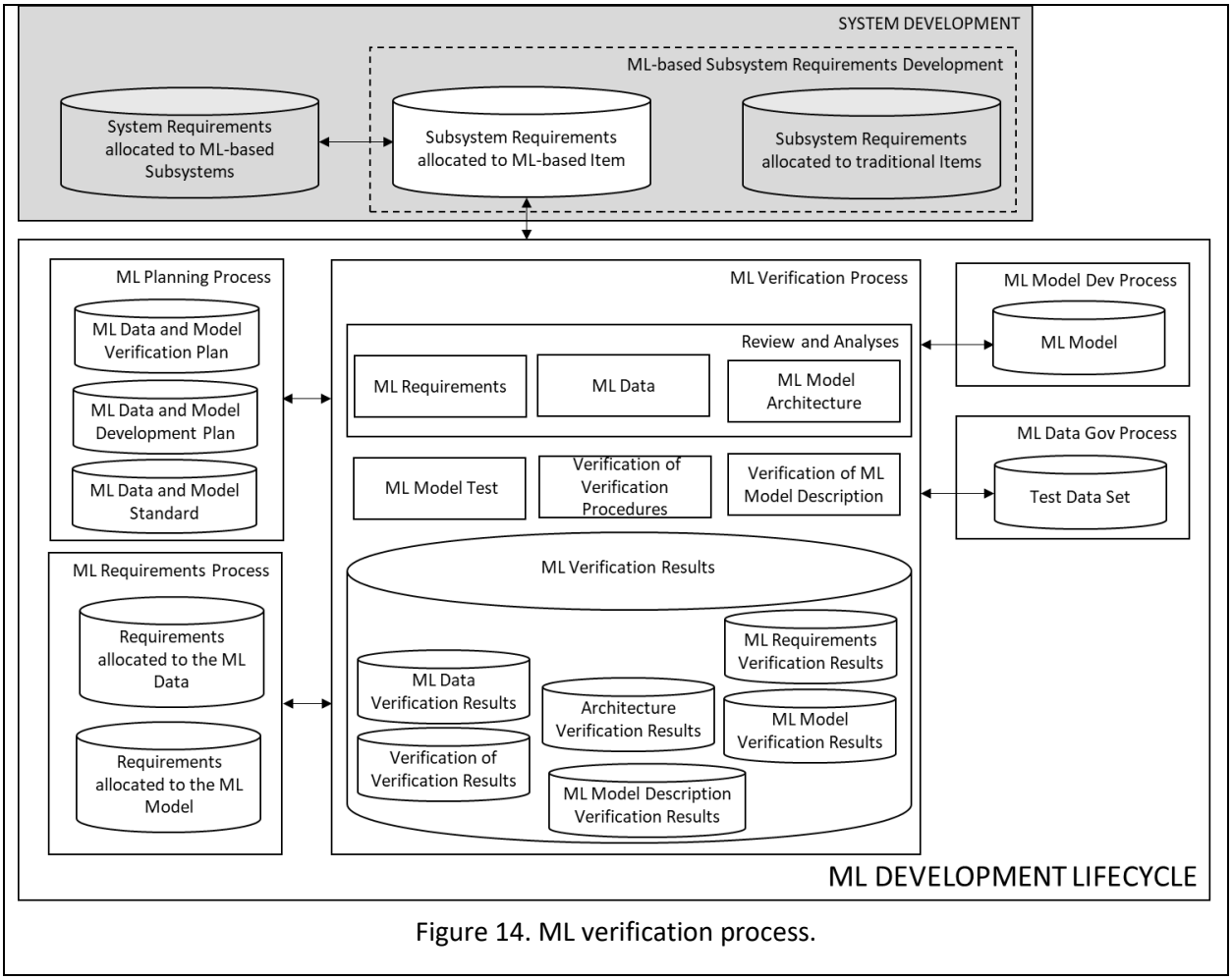

Figure 14. ML verification process.

Section 6.3.1 Purpose of MLDL Verification

The MLDL verification purpose description is similar to that discussed in DO-178C Section 6.1.

The MLDL verification process should identify and report mistakes and errors that occurred during the MLDL. The identification of the mistakes and errors is an important step in the development assurance process. The MLDL verification process will occur across the entirety of the MLDL.

a. The MLDL verification process should confirm the MLDL requirements reflect the systems requirements allocated to the ML-based item.
b. The MLDL verification process should confirm the MLDL requirements were developed into ML data set and ML model requirements that satisfy the systems requirements allocated to the ML-based item.
c. The MLDL verification process should confirm the ML data set reflects the ML data set requirements and is representative of the operational design domain (ODD), including nominal cases, corner cases, edge cases, and adverse cases.
d. The MLDL verification process should confirm the ML model reflects the ML model requirements and is representative of the performance and safety requirements allocated to the ML-based item from the system requirements.
e. The MLDL verification process should confirm the ML architecture is representative of the system requirements allocated to the ML-based item, and the target hardware deployment indicated by the system requirements.
f. The MLDL verification process should confirm the ML concerns and ML recommendations have been appropriately addressed.
g. The MLDL verification process should use appropriate traditional, e.g., requirements base testing, and non-traditional verification methodologies, e.g., adverse stress testing, to ensure the performance of the ML model under all foreseeable required conditions.
h. The MLDL verification process should produce reports that capture deficiencies for the ML data set and the ML model.

Section 6.3.2 Overview of MLDL Verification Process Considerations

The MLDL verification process overview is similar to the approach established in DO-178C Section 6.2.

The MLDL verification process is composed of a series of verification processes. These processes are aligned with the MLDL lifecycle processes, and the program may use the completion of these verification processes as transition criteria. The MLDL verification plan should detail the alignment of the MLDL verification processes with the MLDL processes. Moreover, the MLDL verification plan should detail the various verification techniques used for the completion of each of the ML verification processes. For example, reviews should be used as the MLDL verification technique used to evaluate the adequacy of the MLDL requirements. The review verification technique should be evaluated for adequacy to ensure there will be no oversights when the verification reviews are conducted. For other MLDL verification processes, the use of MLD test cases and procedures may be necessary to fully assess the adequacy of the MLDL process. For example, the MLDL test cases and procedures should be used to evaluate the completion and representativeness of the ML data set against the ML data set requirements. The MLDL test cases and procedures should be reviewed for adequacy, clarity, and consistency with the guidance from the MLDL verification plan, e.g., listing of the environment, inputs, output, and pass/fail criteria. When the MLDL test cases are judged complete, e.g., through a transition criterion, then they should be used to conduct test execution of the ML data set. The MLDL test execution should be rigorous in its execution to identify any mistakes and errors in the completeness, representativeness, and performance. Throughout the execution of the MLDL verification activities, appropriate involvement of quality assurance should be included. Moreover, the MLDL verification activities should also follow the appropriate configuration management process. Finally, any deficiencies should be appropriately reported and adjudicated.

Some of the outputs of the MLDL verification process are the following:

- MLDL Verification Cases and Procedures (see Section 11.15),
- MLDL Verification Results (see Section 11.16), and
- the associated Trace Data (see Section 11.22).

The following are specific MLDL verification considerations, since the MLDL occurs prior to the MLIL:

- To the greatest extent possible, appropriate levels of independence in personnel should be applied between those responsible for the development of the MLDL requirements, those responsible for the development of the ML data set and ML model, and those responsible for the MLDL verification process.
- The MLDL verification of the ML data set should be on constant guard of the ML data set including sim-to-real gaps, and any encounter or concerns uncovered for the ML data set containing any sim-to-real gaps should be noted and adjudicated.
- The MLDL model might not be verified on target deployment hardware, so any hardware gaps between that used for verification and the deployment should be noted and carried forward to the MLIL.
- The MLDL model might not be produced in the deployment model software programming language, so any gap between the MLDL model software programming language and the planned MLIL deployment programming language should be noted and carried forward to the MLIL.
- The MLDL model might not include all the supporting traditional programming elements, so any traditional software programming elements that may be integrated during the MLIL should be noted and carried forward to the MLIL.
- With the concern of catastrophic forgetting associated with ML, the appropriate amount of regression testing should be accomplished to ensure the ML model continues to possess previously present performance and capability.

Section 6.3.3 MLDL Reviews and Analyses

The application of MLDL reviews and analyses process is similar to the approach discussed in DO-178C Section 6.3.

The MLDL verification process will make use of reviews and analyses. The reviews should be guided by a checklist or other similar best practice to ensure formality is brought to the process when applied to the review of the MLDL requirements and the ML data governance and ML model development processes. The analysis process should be traceability driven and have safety implications. For example, traceability of system safety requirements into the ML data set or the ML model. For both approaches, the quality assurance and configuration management processes should be appropriately involved and applied.

Section 6.3.3.1 Review and Analyses of MLDL Requirements

The MLDL requirement reviews and analyses process description is similar to the approach described in DO-178C Section 6.3.2.

The MLDL requirements review, and analyses identify mistakes and errors in the MLDL requirements.

a. The MLDL requirements review, and analyses should ensure the MLDL requirements are in compliance with the system requirements allocated to the ML-based item.
b. The MLDL requirements review, and analyses should ensure the MLDL requirements are accurate and consistent, e.g., ensure the requirements are standardized format and their content is accurate.
c. The MLDL requirements review, and analyses should ensure the MLDL requirements are compatible with the operational design domain identified for the ML-based item by the systems process, e.g., there is no conflict between the MLDL requirements and the operational design domain signals, sources, features, attributes, and scenarios.
d. The MLDL requirements review, and analyses should ensure the MLDL requirements are verifiable, i.e., the MLDL requirements are written such that they can be verified.
e. The MLDL requirements review, and analyses should ensure the MLDL requirements conform to the MLDL requirements standard.
f. The MLDL requirements review, and analyses should ensure the MLDL requirements have bi-directional traceability with the systems requirements allocated to the ML-based item.
g. The MLDL requirements review, and analyses should ensure the MLDL requirements identify the ML algorithm and other necessary algorithms and interfaces to be used.

Section 6.3.3.2 Review and Analyses of MLDL Data Sets

The MLDL data set reviews and analyses process description is similar to the approach described in DO-178C Section 6.3.2.

The purpose of the MLDL data set reviews and analyses process is to identify mistakes and errors that might have occurred during the ML data governance process. The MLDL data set reviews and analyses process will evaluate for the completeness of the ML data set against the ML data requirements and the operational design domain, and check that the ML data set conforms to the ML data standards. The MLDL data set reviews and analyses process will confirm that the ML data set conforms to the following objectives:

a. The review and analyses of the MLDL data set should verify that the MLDL data set complies with the ML data set requirements, including the data hazard analysis.
b. The review and analyses of the MLDL data set should verify that the ML data set complies with the operational design domain.
c. The review and analyses of the MLDL data set should verify that the ML data set is verifiable, and that there are no unverified signals, sources, features, attributes, and scenarios within the ML data set.
d. The review and analyses of the MLDL data set should verify that the ML data set complies with the ML data standards, covers the ML concerns, and addresses the ML recommendations.
e. The review and analyses of the MLDL data set should verify that bi-directional traceability exists between the ML data set and ML data requirements.
f. The review and analyses of the MLDL data set should verify that the ML data set is accurate, complete, and representative of the ML data set requirements and the operation design domain, including robustness data sets which cover the adverse cases, edge cases, and corner cases. For example, the accuracy and consistency objective should determine no errors were introduced during the processing activity, e.g., tagging errors.

g. The review and analyses of the MLDL data set should verify that the ML data set is adequate and sufficient. For example, the objective is to determine that adequate and sufficient data set volume/quantity was collected or created to ensure the model is trained, validated, and tested appropriately, or the simulation environment is of the appropriate fidelity and duration that the ML model can be appropriately trained, tested, and verified.

Section 6.3.3.2.1 MLDL Data Assurance Processes & Methods

Figure 15 presents an approach for conducting a review and analysis of MLDL Data Sets, culminating in the verification of the data set.

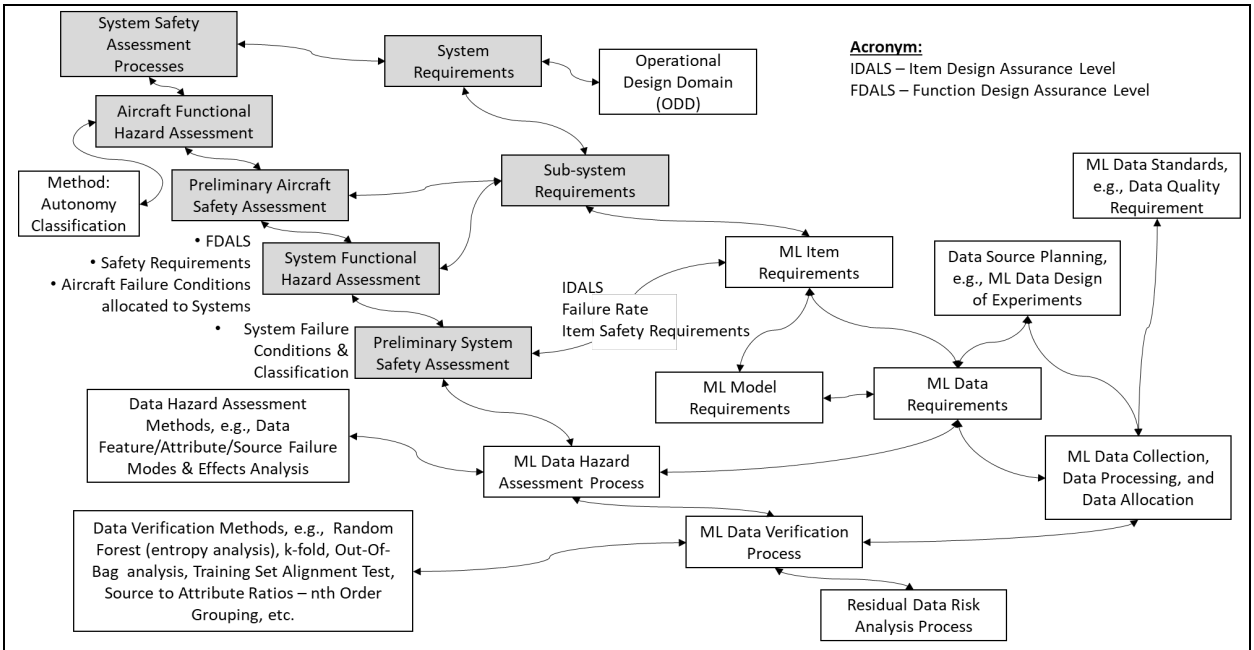

Figure 15. Data assurance assessment processes and methods.

Notes for Figure 15:

- Note 1: Data includes the features, attributes, and classes present in the data as well as the samples, signals, sources, and collection of those features, attributes, and classes.
- Note 2: Data is most applicable to data-driven ML techniques; however, similar techniques more applicable to Reinforcement Learning will be specifically covered in the future, e.g., scenario planning, assessment, and verification.
- Note 3*: Ultimately the Data Governance Process produces data sets, e.g., training, validation, and test, and the ML Data Processing Description, which contains the processes used for the selection collection, preparation, and allocation of the datasets.
- Note 4: ML Data Requirements have dependency upon the ML Model technique and ML Model type.

Section 6.3.3.2.2 Data Hazard Assessment

Data hazard assessment is a hazard assessment process that leverages techniques applied to traditional system and software hazard analysis techniques or introduces novel techniques to assess the hazard impacts associated with data used for ML model development, e.g., training, validation, and testing. In the discussion that follows, the failure modes and effects analysis (FMEA) is offered up as one type of data hazard analysis technique, but others are viable.

As indicated in ARP 4761: "An FMEA is a systematic, bottom-up method of identifying the failure modes of a system, function, or item and determining the effects on the next higher level. It may be performed at any level within the system (e.g., piece-part, function, black box). Software can also be analyzed qualitatively using a functional FMEA approach. Typically, an FMEA is used to identify failure effects resulting from single failures." An additional resource for the application and development of the FMEA is ARP 5580[50].

As shown in Figure 16, for the discussion in this document, the ML Data FMEA is a data hazard analysis technique that would be used to assess the failure effects of the data, e.g., attributes, features, signals and sources, and the hazard effect on the ML model.

[50] Recommended Failure Modes and Effects Analysis (FMEA) Practices for Non-Automobile Applications, issued: 2001-07, Reaffirmed: 2020-08 [URL: https://www.sae.org/standards/content/arp5580].

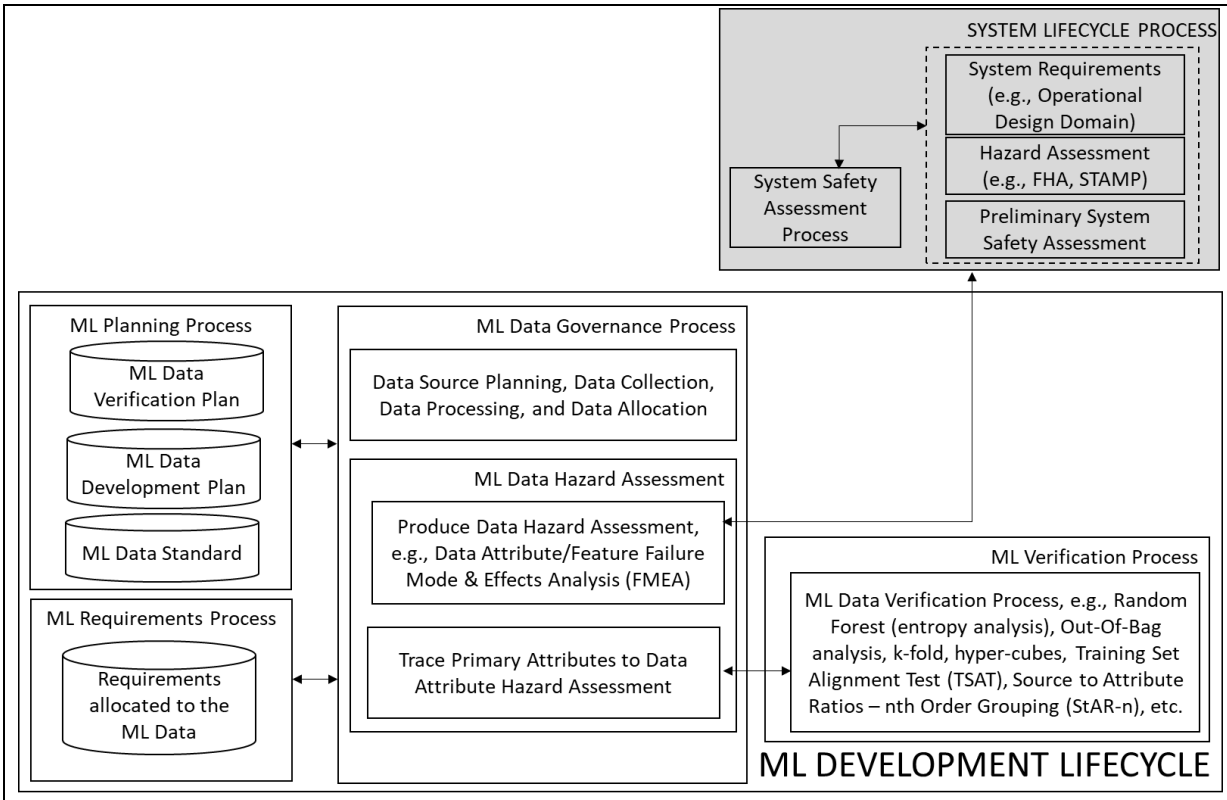

Figure 16. Data hazard assessment.

As shown in Figure 16, inputs to the ML Data FMEA include the following:

- The systems-level hazard assessment artifacts, including the systems' functional hazard assessment which should assess the hazard impacts of the functions within the ML-based system.
- The ML data requirements.
- The ML data source planning, data collection processes, data processing process, and data allocation process, which may include data design of experiments type artifacts indicating what data is necessary and why.

Other artifacts that influence the ML Data FMEA are the following:

- ML Data Development Plan
- ML Data Standard
- ML Data Requirements Standard

Data FMEA is a safety assessment of data features, attributes and sources, samples, and signals anticipated to drive the ML model generalization in its operational design domain. Each of those is analyzed similarly to the approach used by a Software Interface FMEA described in ARP5580. Failure modes include reasons why data may be incorrect, inaccurate, missing, biased, or deficient in one or more data quality attributes, and the effects of those failures. Ultimately, the Data FMEAs may be summarized into a Data Failure Modes Effect Summary (FMES) to support the failure modes analysis

considerations. Given the sensitivity of the ML model to input data sources, developing the data FMEA or its equivalent, is encouraged.

As an example, Table 1, from the Computer Vision- Hazard and Operability Study (CV-HAZOP)[51], shows a data FMEA establishing traceability from failures (errors) in the data attributes, features, samples, sources, and signals to resultant impact of the ML model. The table identifies the source (location) and feature/attribute (guide word) and the consequences and risk.

Table 1. Example data FMEA.[52]

Risk Id	Location	Guide Word	Parameter	Meaning	Consequence	Risk
0	Light Sources	No (not none)	Number	No light sources	No light available	Sensor will receive no light, but thermal noise or black current can cause wrong input
1	Light Sources	More (more of, higher)	Number	Many light sources (more light sources than expected)	Too much light	Overexposure (of whole image)
2	Light Sources	More (more of, higher)	Number	Many light sources (more light sources than expected)	Too few shadows	Algorithms using shadows can be confused
3	Light Sources	Less (less of, lower)	Number	Few light sources (fewer light sources than expected)	Too faint light (in parts of the scene)	Sensor will receive too faint light from some scene regions
4	Light Sources	Less (less of, lower)	Number	Few light sources (fewer light sources than expected)	Too many shadows	Algorithms can be confused by shadows
5	Light Sources	Less (less of, lower)	Number	Few light sources (fewer light sources than expected)	Very sharp shadows	
6	Light Sources	As well as	Number	Mirrors fake additional light sources	Light sources can appear at locations other than where they are	Algorithm confuses position of light sources

Additional steps would be to indicate the expected ML effect, and eventually, if the data is available, the probability of each. Currently the CV-HAZOP has 1,469 entries. Such a systematic approach by computer vision experts allows for the ML-based software item to be appropriately representative and complete data sets to be collected and used for ML training. Data sets used can be assessed against the data hazard assessment to determine if all negative consequences and high risks effects are covered. Where gaps in the data set exist, appropriate mitigations can be determined, e.g., creation of synthetic data, creation of a derived subsystem requirement to mitigate, or others. Any gaps that remain in the

[51] O. Zendel, K. Honauer, M. Murschitz, M. Humenberger and G. F. Domínguez, "Analyzing Computer Vision Data — The Good, the Bad and the Ugly," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6670-6680, doi: 10.1109/CVPR.2017.706, [URL: https://openaccess.thecvf.com/content_cvpr_2017/papers/Zendel_Analyzing_Computer_Vision_CVPR_2017_paper.pdf]

[52] cv-hazop, https://vitro-testing.com/cv-hazop/

data set should be indicated in the ML data processing and MLDL verification output data item and brought to the attention of the airworthiness authority.

Section 6.3.3.2.3 Ensuring Statistically Relevant Coverage of Training Data Sets

Statistical data assessment of ML data sets is an emerging field, so various methods are being discussed as valid depending on the situation. For this document, a few different techniques will be mentioned with appropriate references to guide their application:

- Random Forest, e.g., can be used to help determine feature importance
- Clustering, e.g., can be used to help identify feature redundance[53]
- k-fold cross-validation, e.g., can be used to determine data quantity adequacy
- Training Set Alignment Test[54], which is a design of experiments approach
- Source to Attribute Ratios – nth Order Grouping
- hyper-cubes, e.g., can be used to help determine data completeness[55]
- distribution discriminator framework, e.g., determine if data is out-of-distribution[56]

The ML-based software item developer may have different techniques they prefer. In such a situation the vendor should indicate their selection. The evaluation and justification of the statistical relevance of the data set should be conducted regardless of the approach for determining such validity. The goal of these approaches is to quantitatively show, through statistical analysis, that the data set selected, e.g., samples, signals, sources, attributes, and features, contains a complete representation of the operational design domain. While the method matters, more important is that the processes are pursued. Approaches to present the valid statistical representation may involve the following techniques or others unique to the vendor's approach:

- Exploratory Data Analysis[57]
- Boxing clever[58]
- Datasheets for datasets[59]

These results should present these statistical examinations of the data sets. The results should explain where the data set does not statistically fulfill the requirements associated with the operational design domain.

[53]Clustering, https://github.com/fastai/fastbook/blob/master/09_tabular.ipynb
[54] Training Set Alignment Test, https://dair.nps.edu/handle/123456789/4393
[55] Hypercube, http://iiisci.org/journal/PDV/sci/pdfs/EA039OY16.pdf
[56] "Concepts of Design Assurance for Neural Networks (CoDANN)", EASA and Daedalean AG, AI, Version 1.0, March 31, 2020. [URL: https://www.easa.europa.eu/sites/default/files/dfu/EASA-DDLN-Concepts-of-Design-Assurance-for-Neural-Networks-CoDANN.pdf, Accessed: 10/18/2022]
[57] Data Analysis, Exploratory, David R. Brillinger, University of California, Berkeley, 2011, [URL: https://www.stat.berkeley.edu/~brill/Papers/EDASage.pdf]
[58] Boxing clever: Practical techniques for gaining insights into training data and monitoring distribution shift, R. Ashmore and M. Hill, in First International Workshop on Artificial Intelligence Safety Engineering, 2018, [URL: https://link.springer.com/content/pdf/10.1007%2F978-3-319-99229-7_33.pdf].
[59] T. Gebru, et al., 94 Datasheets for datasets, arXiv, 1803.09010, December 2021, [URL: https://arxiv.org/pdf/1803.09010.pdf].

Section 6.3.3.3 Review and Analyses of MLDL ML Model Architecture

The MLDL ML model architecture reviews and analyses process description is similar to the approach described in DO-178C Section 6.3.3.

The review and analyses of the MLDL ML model architecture is to identify any report and any mistakes and errors that may have occurred during the ML model architecture development process. The review and analyses approach examines the ML model to confirm architecture is bi-directional traceable to the ML requirements and ensure it complies with the ML model and ML data set.

The review and analyses of the MLDL ML model architecture objectives should include the following:

a. The review and analyses of the MLDL ML model architecture should verify that the ML model architecture complies with the ML requirements.
b. The review and analyses of the MLDL ML model architecture should verify that the ML model architecture complies with the ML model.
c. The review and analyses of the MLDL ML model architecture should verify that the ML model is verifiable, and that there are no unverified functions within the ML model.
d. The review and analyses of the MLDL ML model architecture should verify that the ML model complies with the ML standards, covers the ML concerns, and addresses the ML recommendations.
e. The review and analyses of the MLDL ML model architecture should verify that bi-directional traceability exists between the ML model architecture and ML model requirements.
f. The review and analyses of the MLDL ML model architecture should verify that the ML model architecture is robust to adverse cases, edge cases, and corner cases.

Section 6.3.3.4 Review and Analyses of MLDL ML Model

The MLDL ML model reviews and analyses process description is similar to the approach described in DO-178C Section 6.3.4.

The review and analyses of the MLDL ML model is to identify any report any mistakes and errors that may have occurred during the ML model development process. The review and analyses approach examining the ML model to confirm it is bi-directional traceable to the ML model requirements and ensure it complies with the ML model requirements and ML architecture.

The review and analyses of the MLDL ML model objectives should include the following:

a. The review and analyses of the MLDL ML model should verify that the ML model complies with the ML model requirements.
b. The review and analyses of the MLDL ML model should verify that the ML model complies with the ML model architecture.
c. The review and analyses of the MLDL ML model should verify that the ML model is verifiable, and that there are no unverified functions within the ML model.
d. The review and analyses of the MLDL ML model should verify that the ML model complies with the ML model standards, covers the ML concerns, and addresses the ML recommendations.
e. The review and analyses of the MLDL ML model should verify that bi-directional traceability exists between the ML model and ML model requirements.

f. The review and analyses of the MLDL ML model should verify that the ML model is accurate, complete, and representative of the ML model requirements, and the ML model is stable within the operation design domain and robust to adverse cases, edge cases, and corner cases.

The goal of accomplishing these ML model objectives and activities is to establish ML model learning assurance. Learning assurance is a confidence that ML model learning errors are detected and removed, and unexpected, undesired, and erroneous behaviors are identified and adjudicated. Follow-on work will analyze the current learning techniques to determine their adequacy and where enhancements are necessary.

Section 6.3.4 MLDL ML Data and Model Testing

MLDL data and model testing involves confirming that the ML model satisfies the ML data and model requirements. MLDL data and model testing should demonstrate the ML data and model performs its intended function and does not contain any unintended functionality. The MLDL data and model testing should span the full operational design domain from nominal to the edge and beyond to test the ML model robustness, e.g., unavailability of features, attributes, samples, sources, and signals. The output of MLDL data and model testing may indicate that a previous process needs improvement, e.g., additional ML data requirements and data set acquisition are necessary. The ML data and model testing should also consider the use of traditional and non-traditional testing.

Section 6.3.4.1 MLDL ML Data and Model Test Environment

SCSC-153 COM2-6: "The test environment is appropriate." This can be interpreted as meaning that the test environment must adequately represent the operational design domain. This includes nominal, off-nominal, and robustness testing conditions, where the ML model must be able to validate against nominal and erroneous data sets, e.g., features, attributes, samples, sources, and signals. In most cases simulation surrogate representations of real-world conditions will be replicated, where replication level will be proportional to the development assurance level. The higher the development assurance level the higher the expectation of replication. For example, the field-of-view resolution at altitude, vibration, temperature, and atmospheric obstructions for nominal and off-nominal conditions.

For the MLDL, where possible, the target hardware should be used. Discrepancies between the MLDL test environment and the operational design domain, and target hardware should be noted in the ML Verification Results.

Section 6.3.4.2 MLDL ML Data and Model Requirements-based Test Selection

The MLDL requirements-based test selection process description is similar to the approach discussed in DO-178C Section 6.4.2.

ML data and model requirements-based testing follows the tradition established for traditional software, which was found to be effective in uncovering mistakes and errors. The ML data and model requirements-based testing activities are as follows:

a. The ML data and model requirements-based testing should be based on ML requirements, which include the ML data and model requirements.
b. The ML data and model requirements-based testing should include nominal and off-nominal testing, where the nominal includes all applicable scenario and coverage of the operational

design domain, and the off-nominal testing includes robustness scenarios and data sets to create adverse stress test cases and robustness test cases.

c. The ML data and model requirements-based testing should develop the nominal and off-nominal test cases based on the ML requirements.
d. The ML data and model requirements-based testing should use the test cases to generate the test procedures.
e. The ML data and model requirements-based testing should evaluate the conformance to the ML recommendations and ensure the ML concerns are addressed.

The operational design domain and the ML requirements are used to develop scenarios. Those scenarios are used to create the nominal and robustness (including failure) test cases. Moreover, the robustness and coverage analysis testing should be scenario-driven based on the operational design domain specified in the ML requirements. The development of the scenarios is based on the ML requirements (data and model) and the operational design domain, which is also included in the ML requirements. The ML requirements are based on system requirements, which may include the concept of operations (CONOPS). The robustness and coverage testing usage of scenario-driven considerations may need to incorporate CONOPS to fully address the requirements (ML, which are derived from the system).

To reach performance requirements, the requirements-based testing may require the use of test cases that involve the use of randomly selected data subsets of test data executed iteratively over a number of times.

Section 6.3.4.2.1 MLDL ML Data and Model Normal Range Test Cases

MLDL ML data and model normal range test cases verify the data is adequate and the model has adequate performance and stability for the nominal scenarios and portions of the operational design domain. MLDL ML data and model normal range test cases activities include the following:

a. The MLDL data and model normal range test cases should use the ML test data set, where equivalence classes[60] can be established from the test data set.
b. The MLDL data and model normal range test cases should use components of the test data set, which include nominal features, attributes, signals, sources, and scenarios under nominal operational design domain.
c. The MLDL data and model normal range test cases should use the test data set over the full range of values including boundary values, i.e., edge cases and corner cases.
d. The MLDL data and model normal range test cases should involve multiple iterations of the ML Model to check the characteristics of the function in context.
e. The MLDL data and model normal range test cases should be developed to exercise the transitions possible during normal operation.
f. The MLDL data and model normal range test cases should verify the required characteristic generalization, accuracy, and stability under nominal conditions.

[60] Cantone, Anthony S., "Equivalence Class Testing", Systems Safety Engineering Branch Systems Engineering Department APRIL 2020 Naval Air Warfare Center Weapons Division China Lake, Ca 93555-6100 [URL: AD1101958.pdf (dtic.mil)].

g. The MLDL data and model normal range test cases should use the test data set, so the test data set should cover the normal test range.

Section 6.3.4.2.2 MLDL ML Data and Model Robustness Test Cases

Robustness test cases should be the "unhappy path", where features, attributes, signals, and sources are present in off-nominal operational design domain conditions as well as outside of the operational design domain. This should include but not limited to the following conditions:

- Inputs outside the ranges specified in the operational design domain, such as points slightly outside the flight envelope
- Erroneous and robust data sets
- Failures and drop-out of signals, sources, features, and attributes

Robustness test cases should ensure, using the robustness ML data set, that the ML model is not unsafe under any foreseeable operational design domain conditions. Examples of robustness test cases include verifying input data features, attributes, signals, and sources dropouts to the ML model computation are tolerated.

Robustness test cases should include adverse conditions, edge cases, corner cases, and stress test conditions and scenarios.

The activities for the robustness test cases include the following:

a. Off nominal and invalid elements of the test data set should be used to construct equivalence classes. Off-nominal and invalid elements of the data set include off-nominal and invalid features, attributes, signals, and sources under off-nominal and invalid operational design domain.
b. Data set should involve the use of invalid inputs outside of the operational design domain.
c. Robustness test cases should involve multiple invalid iterations of the ML Model to check the characteristics of the function in an invalid context.
d. Robustness test cases should be developed to exercise invalid transitions during invalid operation.
e. Test cases should verify the ML model generalization, accuracy, and stability under off-nominal conditions, e.g., robust conditions.
f. A check should be made to ensure that protection mechanisms for exceeded conditions respond correctly, e.g., run-time assurance monitors.
g. Robustness test cases should use the test data set, so the test data set should cover the robustness test range.

Section 6.3.4.3 ML Data and Model Requirements-Based Testing Methods

Once the nominal and robustness test cases are established based on the ML requirements, the test methods can be set up. Requirements-based tests establish scenarios that are requirements-based and make use of the validation and test data sets. The requirements-based testing methods include the full operational design domain to be encountered and should be fully representative and complete. Test cases consist of the procedures, test plans, scenarios, data set, and ML model.

The primary MLDL requirements-based testing method includes test-based verification[61], where "this activity involves providing test cases to the trained model and checking the outputs against the expected results[62]".

Section 6.3.4.4 ML Data and Model Coverage Testing

Within DO-178C coverage analysis is a two-step process: (1) requirements-based coverage analysis, and (2) structural coverage analysis (i.e., source code and data and control coupling). The approach for conducting requirement coverage analysis remains the same, where test coverage of implementation requirements is addressed during the MLIL. For structural coverage analysis, some updates are necessary.

As argued by Aerospace Vehicle Systems Institute (AVSI) Authorization for Expenditure (AFE) 87[63], which will be known as AFE-87 in the remainder of the document, for ML, despite the meaningfulness of structural coverage analysis changing, especially software source code coverage, and possibly being diluted[64], the structural coverage analysis technique could still have some value[65,66,67,68,69,70]. AFE-87 provides the following caution with respect to coverage analysis on ML:

"Each study defines its own coverage criteria and provides some comparisons to others. However, research on hierarchies of these criteria is missing, and connection to previously existing criteria such as decision coverage is insufficient… The papers reviewed above provide strategies to generate test inputs and demonstrate that it is easy to come up with examples that violate the intended behavior of neural networks. The lack of detailed specification (and thus a testing oracle) still seems to be an open research topic." [AFE-87]

---

[61] Ma, L., et al., Secure deep learning engineering: A software quality assurance perspective. arXiv 2018, arXiv:1810.04538 [URL: https://arxiv.org/abs/1810.04538].
[62] Ashmore, R., et al., Assuring the machine learning lifecycle: Desiderata, methods, and challenges. arXiv 2019, arXiv:1905.04223 [URL: https://arxiv.org/abs/1905.04223].
[63] Final Report, AFE 87 - Machine Learning, AFE 87 Project Members, 7 May 2020, Version 1.0, 87-REP-01, Public, Issued 5 June 2020 [URL: https://avsi.aero/wp-content/uploads/2020/06/AFE-87-Final-Report.pdf].
[64] Li, Z., et al., "Structural Coverage Criteria for Neural Networks Could Be Misleading," ICSE 2019 New Ideas and Emerging Results, Montreal, CA, May 31, 2019,
[65] Lei, et al., "Deepxplore: Automated Whitebox Testing of Deep Learning Systems." Proceedings of the 26th Symposium on Operating Systems Principles. ACM, 2017. [URL: https://arxiv.org/abs/1705.06640].
[66] Tian, Y., et al., "Deeptest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars." Proceedings of the 40th International Conference on Software Engineering. ACM, 2018. [URL: https://arxiv.org/pdf/1708.08559.pdf].
[67] Ma, L. et al. "DeepGauge: Multi-Granularity Testing Criteria for Deep Learning Systems." 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE) (2018): 120-131, [URL: https://arxiv.org/abs/1803.07519].
[68] Sun, Y., Huang, X., Kroening, D., Sharp, J., Hill, M., & Ashmore, R. (2019). Structural Test Coverage Criteria for Deep Neural Networks. ACM Transactions on Embedded Computing Systems, 18(5s), [94]. [URL: https://doi.org/10.1145/3358233].
[69] Liu, Dongyu et al. "DeepTracker: Visualizing the Training Process of Convolutional Neural Networks." ArXiv abs/1808.08531 (2019): n. pag. [URL: https://arxiv.org/abs/1808.08531]
[70] Liu, Mengchen et al. "Towards Better Analysis of Deep Convolutional Neural Networks." IEEE Transactions on Visualization and Computer Graphics 23 (2017): 91-100. [URL: https://arxiv.org/abs/1604.07043]

While the industry is still determining the modulated value and benefit of conducting ML data and model structural coverage, e.g., neuron activation, neuron boundary, and neuron sign/value analysis, this document recommends continuing to conduct the analysis. The vendor is welcome to argue the benefit of equivalent alternative techniques intending to show no unintended functionality is present and quantify the existence of uncovered ML data and ML model structure, e.g., the use of ML data and ML model saliency.

Note: The quantification of neuron activation typically occurs during ML model optimization, which occurs during the MLDL. During ML optimization non-activated neurons are identified and assessed for removal, as removal of neurons can reduce model complexity. Reduction in model complexity leads to fewer neurons, increased speed, and less performance utilization.

Thus, the objectives for test coverage analysis are the following:

a. Test coverage of ML data requirements and model requirements is achieved.
b. Test coverage of ML data and model structure to the appropriate coverage criteria, including both data coupling and control coupling, is achieved.
c. Test coverage of ML data and model nominal and robustness performance is achieved.

Section 6.3.4.4.1 Requirements-Based Test Coverage Analysis
The MLDL requirements-based test coverage analysis process description is similar to the approach described in DO-178C Section 6.4.4.1.

The requirements-based test coverage analysis verifies the ML data set and ML model against the requirements to determine if there is missing or unanticipated functionality. In addition to determining the validity of the ML data set and ML model, this analysis can also be used to determine if there are gaps or deficiencies in the MLDL requirements.

The requirements-based test coverage analysis activities include the following:

a. The requirements-based test coverage analysis activities should use bi-directional trace data to confirm that test cases exist for each MLDL requirement.
b. The requirements-based test coverage analysis activities should include nominal and robustness test cases.
c. The requirements-based test coverage analysis activities should identify any deficiencies with the requirements or with test cases as problem reports.
d. The requirements-based test coverage analysis activities should confirm that all test procedures are traced to test cases.
e. The requirements-based test coverage analysis activities should use the requirements-based test cases to conduct the structural coverage analysis.

Section 6.3.4.4.2 ML Data and Model Structural Coverage Analysis
The purpose of the ML data and model structural coverage analysis is to determine if the ML requirements-based testing leaves any portions of the ML data and model uncovered. The ML data and model structural coverage analysis applies analysis techniques to determine the justification for the area of the ML data and model to not be covered by the requirements-based testing. The ML data and model structural coverage analysis activities include the following:

a. The ML data and model structural coverage analysis should collect the ML data and model structural coverage details when the requirements-based testing is being executed.
b. The ML data and model structural coverage analysis should consider the approach used to develop the data and model. For example, for the data, consideration of the format of the data and any sequential dependencies for full coverage analysis. For example, for the model, consideration of the framework and program language may indicate the presence of library functionality that may not be included.
c. The ML data and model structural coverage analysis should also consider the usage of the ML data set when conducting the ML model structural coverage. For example, the portion of the test data set should be noted and evaluated for adequacy when conducting the structural coverage analysis, e.g., are the features, attributes, sources, signals, and scenarios adequate for the requirements-based testing.
d. The ML data and model structural coverage analysis should develop a resolution to ML data and model structural coverage deficiencies (see Section 6.3.4.4.3).

Section 6.3.4.4.3 Resolution of Structural Test Case Coverage

The MLDL resolution of structural test case coverage process description is similar to the description established in DO-178C Section 6.4.4.3.

The MLDL structural coverage test cases coverage should be analyzed to determine if all results were as expected and resolve those results that were unexpected. For example, the heat map may disclose that certain areas of the neural network were activated during specific scenarios whereas other areas were not activated, or areas where additions to the data set are necessary. Moreover, the coverage may reveal that none of the verification cases used certain data or activated some portions of the neural network. Similar analysis can be conducted for tabular solution methods. These results should be resolved to ensure proper structural coverage is as expected.

Approaches to resolve the structural coverage concerns include the following:

a. Enhanced MLDL verification cases or procedures: The MLDL verification process requires additional cases, procedures. Each of these may need to be augmented in order to address ML data and model structural coverage gaps.
b. Enhanced MLDL requirements: The MLDL requirements process lacks appropriate requirements to capture the ML data set and the ML model behavior and functionality that is present within the ML model.
c. Enhanced ML data set: The ML data set lacks appropriate samples, sources, signals, features, attributes, scenarios, or fidelity in the simulation environment.
d. Refine ML Model: Model may have need to be quantized, pruned, or refined, e.g., those neuron values are not contributing.

Section 6.3.4.5 Reviews and Analysis of Test Cases, Procedures, and Results

The MLDL reviews and analysis of test cases, procedures, and results process description is similar to the approach described in DO-178C Section 6.4.5.

The review and analysis of test cases, procedures, and results should verify the MLDL testing satisfies the following objectives:

a. The review and analysis of the test cases, procedures and results is to ensure the test coverage of the MLDL requirement is achieved, so the test cases should be correct.
b. The review and analysis of the test cases, procedures and results is to ensure the test cases were developed into test procedures, and the test procedures are correct.
c. The review and analysis of the test cases, procedures and results is to ensure the test results are correct, with respect to the requirements, test cases, and test procedures.
d. The review and analysis of the test cases, procedures and results is to is to ensure that the test results include an explanation of the output of ML data and model performance and behavior, especially when behavior does not align with ML requirements.
e. The review and analysis of the test cases, procedures and results is to ensure that test cases not able to be executed in the MLDL test environment should be indicated in the ML Verification Results. For those test cases, coordination with MLIL should occur to establish a mitigation plan for addressing those test cases in the MLIL or subsystem verification activities.

Section 6.3.5 ML Verification Process Traceability

The ML verification process traceability is similar to that detailed in DO-178C Section 6.5. Moreover, the traditional software elements of the ML-based item will follow the details of DO-178C Section 6.5.

The MLDL traceability verification activities confirm the following:

a. The MLDL trace includes bi-directional traceability from the systems requirements to the ML-based item requirements, and the ML-based item requirements to the ML data requirements and ML model requirements.
b. The ML data set requirements have traceability to the data hazard assessment and the ODD, and the ML data set requirements address the data hazard assessment and cover the ODD.
c. The ML model requirements have traceability to the appropriate requirements allocated to the ML-based item, which should include appropriate performance and safety expectations.
d. ML data sets are bi-direction traceable to ML data requirements.
e. ML model and ML model architecture are bi-direction traceable to ML requirements and ML model requirements.
f. MLDL test cases and procedures are bi-direction traceable to ML requirements, i.e., ML data test cases and procedures are bi-direction traceable to ML data requirements and ML model test cases and procedures are bi-direction traceable to ML model requirements.

Note: for supporting traditional software elements within the ML-based item, the traditional software element high-level requirements may trace to the ML requirements or to systems requirements allocated to the ML-based item.

Section 6.3.6 Verification of ML Parameter Data Items

Should a parameter data item, e.g., JavaScript Object Notation (JSON), YAML Ain't Markup Language (yaml), or configuration (cfg) files, be used to host the ML model hyperparameter and other

configuration characteristics of the ML model during the MLDL, the ML model and parameter data item should be verified to have the following features:

- The ML model has been developed and verified by range testing to correctly handle all Parameter Data Item Files that comply with their defined structure and attributes.
- The ML model is robust with respect to Parameter Data Item Files structures and attributes.
- All behavior of the ML model resulting from the contents of the Parameter Data Item File can be verified.
- The structure of the MLDL output data items allow the parameter data item to be managed separately, i.e., configuration management.

Unless all conditions above are met, parameter data items should not be verified separately from the rest of the ML model.

The objectives identified below apply for parameter data items that will be used to construct the MLDL ML model description. The objectives below may be achieved by a combination of tests, analyses, and reviews.

a. The Parameter Data Item File should be verified to comply with its structure as defined by the ML model requirements; this verification includes ensuring that the Parameter Data Item File does not contain any entries not defined by the ML model requirements. Each data entry in the Parameter Data Item File should also be shown to have the correct value, be consistent with other data entries, and comply with its attributes as defined by the ML model requirements. Note: For certain data entry, their attributes may be the only aspect that needs to be verified.
b. All entries of the Parameter Data Item File have been covered during verification. In other cases, the value of the data entry may need to be verified. If changes are made to the structure or attributes of the parameter data item, then the need for modification and reverification of the ML model should be analyzed.

Note: during the MLIL, a separate parameter data item could be uniquely created based on the input from the ML model description. The MLDL parameter data item should be separate from the MLIL parameter data items. The transition from MLDL parameter data item values to MLIL parameter data item values could occur through the ML model description. The parameter data item objectives will be appropriately verified during the MLIL.

Section 7 Machine Learning Implementation Lifecycle (MLIL) Process

The Machine Learning Implementation Lifecycle (MLIL) aims to produce the ML inference model, which is deployed on the target hardware and fieldable. In addition, the MLIL will produce the output data items to accompany the deployable inference model.

The purpose of this section is to describe the Machine Learning Implementation Lifecycle (MLIL), which includes the addition of the inference ML model implementation (optimization) and verification process in the Software Lifecycle Process. Figure 17 depicts the interaction with and transition from the Machine Learning Development Lifecycle (MLDL) to the MLIL. Highlighting this interaction with the MLDL, Figure 17 is limited in scope, so it does not provide a listing of processes executed during the MLIL. In addition, hardware aspects of integration such as partitioning, memory, throughput, or communication management are not addressed in this document, so will be addressed in follow-on activities. Figure 17 only shows those MLIL processes (Software Lifecycle Processes) impacted by ML processes. For a more comprehensive high-level illustration of the MLIL (Software Lifecycle Processes) refer to Figure 5 "Information Flow Between System and ML Lifecycles." Figure 17 illustrates the necessary artifacts to perform transition from MLDL to MLIL are the following: ML Requirements, ML Model Description, ML Data Processing Description, MLDL Verification Results, and Test Data Set.

The following MLIL sections only discuss those processes of the processes impacted by the addition of ML consideration. In general, the DO-178C Software Lifecycle Process is executed in full for the MLIL. In Figure 17, white fill indicates new ML processes added to the existing DO-178C software lifecycle processes, which are shown with grey fill. The MLIL makes extensive use of DO-178C, so DO-178C should be referenced for the full listing of the Software Lifecycle Processes.

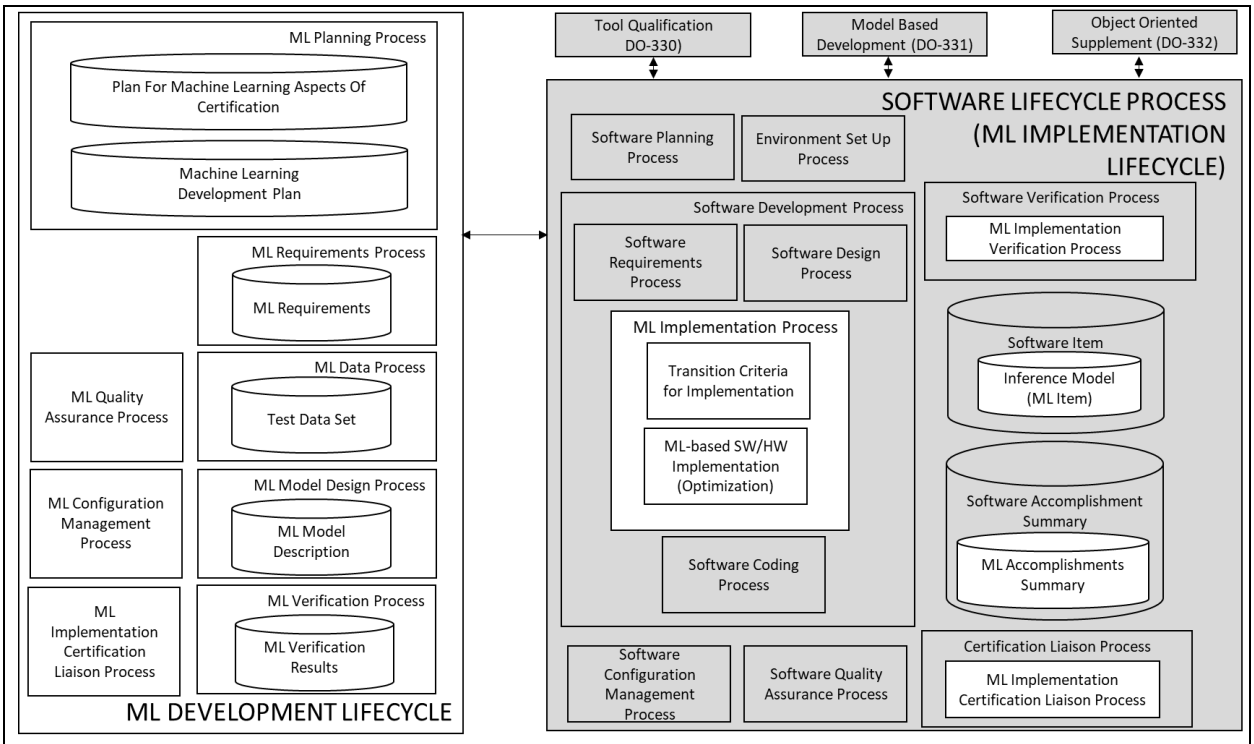

Figure 17. ML implementation lifecycle.

As discussed previously the appropriate coordination of MLDL and MLIL processes should also occur prior to beginning the MLIL Development Process:

- Planning Aspects of Certification Coordination (see Section 4)
- Requirement Processes Coordination (see Section 5)
- Configuration Management Processes Coordination (see Section 8)
- Quality Assurance Processes Coordination (see Section 9)
- Certification Liaison Processes Coordination (see Section 10)

The processes listed above can be executed in a unified approach, i.e., with the MLDL processes and the MLIL process organized and executed jointly or separately, e.g., separate companies executing the MLDL and MLIL. Where necessary, e.g., in cases where separate companies are executing the MLDL and the MLIL, the MLDL planning process, quality assurance, configuration management, and certification liaison output data items may be shared with the MLIL. Moreover, the planning materials should capture a transition plan from the MLDL to the MLIL. Within MLDL to MLIL the transition plan details the responsible parties, appropriate transition criteria, roles, and responsibilities. Differences between the MLDL environment and MLIL environment should be addressed, and any adjudication to account for those differences.

The MLDL outputs, i.e., ML requirements, ML data description, and ML model description serve as an input to the MLIL software requirements process. The MLIL should proceed forward executing the typical software item development assurance process. DO-178C and appropriate airworthiness regulatory guidance should be consulted for any issues related to partitioning, multi-core CPUs, field-programmable gate arrays, etc. Moreover, follow-on ML guidance will pursue necessary unique guidance for ML deployment hardware concerns, e.g., GPUs.

Section 7.1 MLIL Development Process

The MLIL development process builds on the existing Software Development process detailed in DO-178C Section 5.0. That is, the ML Implementation Processes are added to the existing Software Development process. The ML Implementation Processes extend the existing Software Development processes to account for the unique aspects of implementing the ML inference model. In general, those newly added extensions are using the ML model description to code the ML inference model and the test data set to verify the inference model.

Two development objectives added to the existing software development process that are unique to the MLIL process are the following:

a. MLIL inference model is developed, using exact semantic and numerical replication, from the ML requirements and the descriptions, i.e., ML model description and ML Data Processing Description.
b. MLIL inference model deviations, e.g., optimizations, from the ML requirements and descriptions are documented, i.e., justified. Deviations may include numerical deviations in replication, where error bounds would need to be evaluated and justified, or semantic deviations due to implementation adjustments, where more extensive comparative evaluation and justification is necessary.

During the MLIL development process, in the unlikely event that different elements of the ML model are implemented on different processors, a deviation investigation is necessary compare the impact of the MLIL inference model generalization to the MLDL ML verification results. Appropriate justification and performance acceptance evaluation are necessary to ensure the MLIL implementation fulfills the ML model requirements and ML model description.

Section 7.1.1 ML Implementation Process

The primary entrance transition criterion for starting the ML Implementation process is process coordination, as described above of planning, requirements, configuration management, quality assurance, and certification liaison. The following is the minimal set of MLDL output artifacts that should be available to transition into the ML Implementation Process from the MLDL:

- ML Requirements, including the ML data and ML model requirements
- ML Model Description
- ML Data Processing Description
- MLDL Verification Results
- Test Data Set

Additional output MLDL artifacts could include other holdout data sets necessary for the ML inference model verification.

Upon satisfying the transition criteria, the MLIL implementation process may conduct additional optimization of the ML model description or proceed to the ML software coding to follow the exact replication of the ML requirements and the ML model description.

Section 7.1.2 MLIL Coding Process

The MLIL coding process requires robust software development programming languages intended for aviation applications. Moreover, the MLIL software development programming languages may need to execute within a flight and safety-critical real-time operating system (RTOS) environment. These MLIL programming language requirements will eliminate some of the programming languages that may have been used during the MLDL. For example, C++20 [International Organization for Standardization (ISO)/ International Electrotechnical Commission (IEC) 14882:2020] may be more appropriate for the MLIL coding process, whereas Python (www.python.org) could be appropriate for MLDL.

During the MLIL coding process, if additional optimization (e.g., parallelization) and refinement (e.g., transformation) of the ML model is necessary, change impact analysis should be conducted. Those change impacts will be difficult to determine without performing verification of the modified model. Moreover, different programming languages and development techniques could further complicate comparison without performing model verification. Given this, the change impact analysis should be performed by comparing the MLIL inference model verification results with those from the MLDL ML model verification. Should deviation exist from previous verification results, those should be examined to determine if the behavior is still within compliance with the ML requirements or if mitigations are necessary. Mitigation approaches may require an iterative cycle with the appropriate MLDL process to update the MLDL ML model, re-execute the ML verification, and update the ML model description.

MLIL model optimization and refinement performed during the MLIL coding process should be appropriately documented, e.g., as an addendum to the ML model description. Assessment of these or

other deviations should be properly assessed via MLDL/MLIL verification comparison, e.g., behavior and performance.

Section 7.1.3 MLIL Integration Process
Guidance for the MLIL integration process can be found in DO-178C Section 5.4 Integration Process.

Section 7.2 MLIL Verification Process
The MLIL verification process should consist of comparing the inference model verification results with those from the ML model verification. The comparison should execute, at a minimum, the same nominal and robustness tests scenarios used during the ML model verification with the appropriate previously holdout data set. MLIL may need to add additional test scenarios and cases that were not able to be executed during the MLDL model verification process, but necessary to provide coverage of the ML requirements (data and model).

The MLIL verification process should also account for changes in the test environment, where the MLIL test environment should be more representative of the deployment target than the MLDL test environment.

Three verification objectives added to the existing software verification process, that are unique to the MLIL process are the following:

a. MLIL Implementation transition criteria are achieved.
b. MLIL inference model verified against the ML requirements and descriptions.
c. MLIL model deviations from the ML requirements and descriptions are verified.

Section 7.2.1 Purpose of MLIL Verification
The purpose of MLIL verification is to ensure that the inference model has been implemented correctly, i.e., matches the ML Model Description and therefore meets the accuracy, generalization, stability, and robustness guarantees as verified by the MLDL process.

Moreover, as indicated by EASA: “The applicant should provide an analysis on the robustness (or stability) of the inference model.” [EASA IMP-08] The robustness (stability) verification of the inference model should include verification in adverse conditions.

Guidance for the MLIL Verification process can be found in DO-178C Section 6.1 Purpose of Software Verification.

Section 7.2.2 Overview of MLIL Verification Process Activities
Guidance for the Overview of the MLIL Verification process can be found in DO-178C Section 6.2 Overview of Software Verification Process Activities.

Section 7.2.3 MLIL Reviews and Analyses
Guidance for the MLIL Reviews and Analyses process can be found in DO-178C Section 6.3 Software Reviews and Analyses.

Section 7.2.4 MLIL Testing
Guidance for the MLIL Testing process can be found in DO-178C Section 6.4 Software Testing.

Section 7.2.4.1 MLIL Test Environment

Guidance for the MLIL Test Environment can be found in DO-178C Section 6.4.1 Test Environment.

The MLIL Test Environment may differ from the MLDL Test Environment. For example, the MLDL test environment may consist of servers or computers with GPU support, while the MLIL may be more representative of the target deployment hardware, e.g., CPU device with limited computation power and a more restrictive RTOS, which could lead to computation differences affecting the ML inference model. In addition to physical differences between the MLDL and MLIL test hardware, the maturity, diversity, and characteristics of the input test signals available in the MLIL test environment may differ. Each difference between the MLDL and the MLIL test environment should be captured and accounted for in the MLIL tests execution. The amount of regression verification activities should be determined based on the number of differences and their impact on inference ML model performance. Ultimately, the inference ML model is deployed, so confirming no unintended behaviors emerged due to transitioning to the target hardware and environment is critical.

Section 7.2.5 MLIL Verification Traceability

The MLIL traceability process ensures that all ML-based software item requirements are traced and covered by MLIL test cases.

The MLIL Verification Traceability guidance can be found in DO-178C Section 6.5 Software Verification Process Traceability.

Section 7.2.6 MLIL Parameter Data Items

Guidance for the MLIL Parameter Data Items can be found in DO-178C Section 6.6 Verification of Parameter Data Items.

Section 8 ML Lifecycle Configuration Management Process

The ML lifecycle configuration management process consists of the MLDL configuration management process as well as the MLIL configuration management process.

Section 8.1 MLDL Configuration Management Process

The MLDL configuration management process approach is similar to that established in DO-178C Section 7.0. This section covers the objectives and activities for the MLDL configuration management process. The MLDL configuration management process should be executed in accordance with the MLDL planning process (see Section 4) and MLDL Configuration Management Plan (see Section 11.4).

The MLDL configuration management process detailed in the ML planning process (see Section 4) and the MLDL configuration plan (see Section 11.4) should cover the objectives and activities detailed in the following.

The MLDL configuration management process outputs include the MLDL Configuration Management Records (see Section 11.19), MLDL Problem Reports (see Section 11.17), MLDL Configuration Index (see Section 11.14), and the MLDL Environment Configuration Index (see Section 11.13).

The MLDL configuration management process must be aligned with the other process of the MLDL:

- The MLDL configuration management process must be engaged at the beginning of the MLDL and active throughout the MLDL.
- The MLDL configuration management process must provide a process that allows for the retrieval of MLDL requirements, MLDL data sets, ML model, and other program output data items to support assurance assessments, mishap investigations, and program reuse.
- The MLDL configuration management process must ensure the revision control of created, modified, updated, and deprecated artifacts.
- The MLDL configuration management process must establish baselines that are useful during transition criteria for MLDL processes.
- The MLDL configuration management process must ensure changes occur only after configuration review board approval occurs and no unapproved changes or revisions occur.
- The MLDL configuration management process must ensure the revision control is under appropriate back up processes.

Section 8.1.1 MLDL Configuration Management Process Objectives

The MLDL configuration management (CM) process objectives description is similar to the approach described in DO-178C Section 7.1.

The MLDL configuration management process objectives are the following:

a. The MLDL configuration management process should provide the revision and configuration identification of the MLDL output data items and other program artifacts.
b. The MLDL configuration management process should provide the approach for establishing MLDL baseline artifacts, which are used in subsequent MLDL processes.
c. The MLDL configuration management process should include developing problem reports that capture deficiencies and gaps between the MLDL and the MLDL plans and standards.

d. The MLDL configuration management process documents the change, revision, impact, and considerations for the revision.
e. The MLDL configuration management process should use configuration control board processes, which include change impact analysis, to ensure all modification revisions are approved prior to the modifications.
f. The MLDL configuration management process should provide output artifacts to ensure the process is proceeding as necessary in accordance with the ML configuration management plan, e.g., baselines, problem reports, etc.
g. The MLDL configuration management process should use a revision process where the MLDL artifacts are retrievable, especially in the event a rollback is needed, or a catastrophic event occurs.
h. The MLDL configuration management process should provide configuration and revision control of the MLDL tools and environment – including configuration and revision control of the revision control tool.

These MLDL configuration management objectives should be accomplished regardless of development assurance level.

Section 8.1.2 MLDL Configuration Management Process Activities

The MLDL configuration management process activities description is similar to the approach described in DO-178C Section 7.2.

The MLDL configuration management process establishes activities for the assignment of revision identification and configuration identification. This revision identification should be used to track the configuration identification of the MLDL output data item. The final revision identification should be transitioned from the MLDL to the MLIL and provided to the certifying authority to establish the baseline for the ML-based item. The revision identification can be used to track and trace to the changes that occurred for a specific version of the ML-based item.

Section 8.1.2.1 MLDL Configuration Identification

The MLDL configuration management identification description is similar to the approach described in DO-178C Section 7.2.1.

The MLDL configuration identification process includes the following:

a. The MLDL configuration activities should establish the revision identification of the MLDL output data items, and other program artifacts, e.g., transition criteria.
b. The MLDL configuration activities should establish the revision identification for aggregates or subsets of MLDL output data items or other program artifacts, e.g., the ML data set may have unique version identification for the training, validation, and test data sets, and then an overall revision number for the ML data set.
c. The MLDL configuration activities should establish the revision identification prior to the use of the ML output data items in subsequent processes, e.g., the ML data set should be under configuration management and have a revision number prior to being used for ML model training, validation, and testing.

Section 8.1.2.2 MLDL Data Set and ML model description Baselines and Traceability

The following baseline and traceability guidance is derived from DO-178C Section 7.2.2.

The MLDL data sets and ML model description baseline activities include:

a. During the MLDL, data set baselines, which can consist of training, validation, and test data sets, should be established for ML configuration items used for certification credit. Intermediate data set baselines may be established to aid in iterative development, but the data set that leads to the development of the ML model description should be baselined and controlled throughout the ML-based software item lifecycle process activities.
b. An ML model description baseline should be established for the ML model description defined in the ML Configuration Index (see Section 11.14).
c. Data set and ML model description baselines should be established in controlled libraries, whether physical, electronic, or other, to ensure their integrity. Once a baseline is established, it should be protected from change.
d. Change control activities should be followed to develop a derivative baseline from an established baseline.
e. If certification credit is sought for ML lifecycle process activities or data associated with the development of the previous baseline, then a baseline should be traceable to the baseline from which it was derived.
f. If certification credit is sought for ML lifecycle process activities or data associated with the development of the previous configuration item, a configuration item should be traceable to the configuration item from which it was derived.
g. A baseline or configuration item should be traceable either to the output it identifies or to the process with which it is associated.

Section 8.1.2.3 MLDL Problem Reporting, Tracking, and Corrective Action

The MLDL problem reporting, tracking, and corrective action description is similar to the approach described in DO-178C Section 7.2.3.

The MLDL problem reporting, tracking, and corrective action activities include:

a. The MLDL problem reporting, tracking, and corrective action activities should describe discrepancies and deficiencies with the artifact against the MLDL plans, MLDL standards, or MLDL requirements.
b. The MLDL problem reporting, tracking, and corrective action activities should include revision identification or other configuration identification in the problem report.
c. The MLDL problem reporting, tracking, and corrective action activities should indicate when the change control activity review is necessary due to change control discrepancies or deficiencies.

Section 8.1.2.4 MLDL Change Control

The MLDL change control description is similar to the approach described in DO-178C Section 7.2.4.

The MLDL change control activities include:

a. The MLDL change control activities should provide configuration management control of the artifacts under revision control, where the integrity of the revision control is maintained.

b. The MLDL change control activities should revise the configuration identification when the revision is updated.
c. The MLDL change control activities should ensure revisions coordinated appropriately with change control boards or revision authority, and all revisions are traceable to approved problem reports or revision requests.
d. The MLDL change control activities should ensure change impact analysis is appropriately considered for any and all revisions that occur.
e. The MLDL change control activities should ensure changes are appropriately recorded in plans, reports, and other appropriate artifacts.
f. The MLDL change control activities should ensure all processes are using the appropriate versions of the artifacts, e.g., ML model training is using the appropriate version of the ML data set.
g. The MLDL change control activities should ensure the tool and environment configuration are appropriately under revision control.
h. The MLDL change control activities should ensure all changes are tracked to a revision control approval process.

Section 8.1.2.5 MLDL Change Review

The MLDL change review is similar to the approach established in DO-178C Section 7.2.5. The MLDL change review process should accomplish the following:

a. The MLDL change review process should institute a configuration change board or similar to adjudicate changes.
b. The MLDL change review process should adjudicate based on the change being an error or mistake or a missed requirement, data set element, or ML model architecture design.
c. The MLDL change review process should provide resultant derived requirements, which address changes, to the systems process for consideration.
d. The MLDL change review process should indicate those involved in the problem review, e.g., quality assurance, data analyst, etc.
e. The MLDL change review process should indicate the resolution action to be taken.
f. The MLDL change review process should indicate the status of the resolution taken.
g. The MLDL change review process should trace changes to the Problem Reports.

Section 8.1.2.6 MLDL Configuration Status Accounting

The MLDL configuration status accounting approach is similar to that established in DO-178C Section 7.2.6. The MLDL configuration status accounting activities include the following:

a. The MLDL configuration status accounting should be able to report the configuration of all output data items, which include ML plans, ML standards, ML data set, ML model, ML Data Processing Description, ML model description, etc.
b. The MLDL configuration status accounting should be appropriately accessible for review by stakeholders, e.g., quality assurance should be able to access the configuration management status and review for completeness and quality.

c. The MLDL configuration status accounting should ensure the configuration management is appropriately backed up for interruption events.
d. The MLDL configuration status accounting should have traditional characteristics such as baseline identification and status.

Section 8.1.2.7 MLDL Archive, Retrieval, and Release

The MLDL archive, retrieval, and release approach is similar to that established in DO-178C Section 7.2.7. The MLDL configuration management archival, retrieval and release should consist of the following:

a. The MLDL configuration management retrieval process should include appropriate checks and balances to ensure the retrieved ML data or ML model version is identical to the archived version, e.g., CRCs, MD5, etc.
b. The MLDL configuration management archival process should store the ML data set and ML model for a time period established by the program in order to support updates and mishap investigations.
c. The MLDL configuration management transition process should ensure the version artifacts transitioned to the MLIL will not be corrupted through the transmissions and initial access process.
d. The MLDL configuration management archival process should appropriately back up the program configuration management during the ML lifecycle.

Section 8.1.3 MLDL Data Control Categories

MLDL lifecycle data can be assigned to one of two configuration management control categories: Control Category (CC) 1 (CC1) and Control Category 2 (CC2). Table 2 defines the set of MLDL configuration management process activities associated with each control category, where the minimum activities apply to that category's MLDL data. CC2 activities are a subset of the CC1 activities.

Table 2. MLDL CM process activities associated with CC1 and CC2 data.

MLDL Configuration Management Process Activity	Reference	CC1	CC2
MLDL Configuration Identification	8.1.2.1	●	●
MLDL data set and model baselines	8.1.2.2.a 8.1.2.2.b 8.1.2.2.c 8.1.2.2.d 8.1.2.2.e	●	
MLDL Traceability	8.1.2.2.f 8.1.2.2.g	●	●
MLDL Problem Reporting	8.1.2.3	●	
MLDL Change Control–- integrity and identification	8.1.2.4.a 8.1.2.4.b	●	●

MLDL Configuration Management Process Activity	Reference	CC1	CC2
MLDL Change Control–- tracking	8.1.2.4.c 8.1.2.4.d 8.1.2.4.e 8.1.2.4.f 8.1.2.4.g 8.1.2.4.h	●	
MLDL Change Review	8.1.2.5	●	
MLDL Configuration Status Accounting	8.1.2.6	●	
MLDL Retrieval	8.1.2.7.a	●	●
MLDL Protection against Unauthorized Changes	8.1.2.7.b	●	●
MLDL Recreation	8.1.2.7.b	●	
MLDL Transition	8.1.2.7.c	●	
MLDL Archival	8.1.2.7.d	●	●

The Annex A tables specify the control category by software level for the software life cycle data items.

Section 8.1.4 MLDL Load Control

Load control ensures that the item, and if necessary, parameter data item files, are loaded into the system or equipment with appropriate safeguards. Load control is not applicable to MLDL. Load Control will be established during the MLIL following the guidance of DO-178C Section 7.4.

Section 8.1.5 MLDL Environment Control

The MLDL environment control approach is similar to that established in DO-178C Section 7.5.

The MLDL environment control is focused on controlling the tools used for the creation, development and modification of the ML data set and ML model. The activities are the following:

a. The MLDL Environment Control should identify through version control and configuration management the tools and tool configurations for the creation, development, and modification of the ML data set.
b. The MLDL Environment Control should identify through version control and configuration management the tools and tool configurations for the creation, development, and modification of the ML model.
c. The MLDL Environment Control should identify through version control and configuration management the verification tools and tool configurations for the verification of the ML data set and ML model.
d. The MLDL Environment Control should comply with the appropriate configuration management category, i.e., Category 1 or Category 2.
e. The MLDL Environment Control should assign Category 1 to 8.1.5.a and 8.1.5.b and Category 2 to 8.1.5.c.

Section 8.2 MLIL Configuration Management Process

The guidance provided by DO-178C Section 7.0 for configuration management remains applicable for MLIL.

Section 9 ML Lifecycle Quality Assurance Process

The ML lifecycle quality assurance (QA) process consists of the MLDL quality assurance process and the MLIL quality assurance process.

Section 9.1 MLDL Quality Assurance Process

The MLDL quality assurance process is similar to the approach established in DO-178C Section 8 but instead applied to the MLDL and not just the MLIL.

The primary goal of the MLDL quality assurance, as established in the ML Quality Assurance Plan (see Section 11.5), is to establish a conformity approach for the MLDL process and ensure the MLDL conformity approach is being followed. The quality assurance process for the MLIL is covered in DO-178C Section 8. Establishing the MLDL conformity review approach should be established such that it ensures the MLDL planning process through to the MLDL verification process and transition to the MLIL process is fully evaluated. Additional quality assurance checklists and sampling are also encouraged to ensure quality and completeness is present in the execution of the MLDL quality assurance process. The execution of the quality assurance process should involve quality assurance participation in appropriate events and milestones for witnessing. The outputs of the MLDL quality assurance process are the Machine Learning Quality Assurance Records (see Section 11.20).

Given the novelty of the data assurance and ML model design process ensuring appropriate quality and completeness of these processes is essential. All deficiencies in these processes should be appropriately identified, reported, and adjudicated.

If beneficial the MLDL and the MLIL quality assurance programs can be combined but that should be addressed in the ML Quality Assurance Plans.

Section 9.1.1 MLDL Quality Assurance Objectives

The MLDL quality assurance objectives are the following:

a. MLDL plans, standards, process, objectives, activities, and output data items are developed appropriately.
b. Assurance is obtained that MLDL processes comply with plans and standards.
c. Assurance is obtained that transition criteria for the MLDL processes are satisfied.
d. A conformity review of the MLDL artifacts is conducted.

Section 9.1.2 MLDL Quality Assurance Activities

The MLDL quality assurance activities are similar to those indicated in DO-178C Section 8.2. The full list of quality assurance activities is provided there for the MLIL process. The list of MLDL quality assurance activities includes the following:

a. The MLDL quality assurance process should ensure proper conformance to the recommendations.
b. The MLDL quality assurance process should ensure all deficiencies are reported as problem reports and conformity report findings.
c. The MLDL quality assurance process should ensure all derived ML requirements, derived ML data requirements, or derived ML model requirements are flowed back to the systems process.

d. The MLDL quality assurance process should ensure all safety assessment inputs to the MLDL process are appropriately adjudicated, e.g., data hazard assessment is adjudicated through the development of the ML data requirements and ML data set.
e. The MLDL quality assurance process should witness the ML verification process events.
f. The MLDL quality assurance process should attend the key transition milestone events and document then accordingly in the conformity report.
g. The MLDL quality assurance process should ensure traceability is accomplished.
h. The MLDL quality assurance process should ensure MLDL verification of the ODD and ML model performance is in accordance with plans and procedures.
i. The MLDL quality assurance process should ensure the configuration management of the ML data set and ML model is in accordance with CM plans.

Section 9.2 MLIL Quality Assurance Process

The guidance provided by DO-178C Section 8.0 for quality assurance remains applicable for MLIL.

Section 10 ML Lifecycle Certification Liaison Process

The ML Lifecycle Certification Liaison Process involves the MLDL Certification Liaison Process, and the MLIL Certification Liaison Process. A recommended approach for establishing the ML Lifecycle Certification Liaison Process involves addressing expectations for both at the outset of the ML lifecycle. While this is the recommended approach, another approach is to address them separately. If they are treated separately coordination between the two processes must be established and maintained.

In general, the goal of the certification liaison process is to achieve certification/approval, which is obtained by providing the airworthiness authority with appropriate explanations on how the system works and the verification evidence that it meets its requirement to accept it. The following provides additional details on establishing that process and executing it successfully.

Due to the novel nature of the use of ML, coordination with the certification authority should be established as soon as possible. Coordination will begin by discussing the project overview and reviewing the plan for aspects of certification (see Section 4).

Section 10.1 Overview of MLDL Certification Liaison Process

As with DO-178C Section 9.0, the objectives of the certification liaison process for MLDL are the following:

a. Establish communication and understanding between the applicant and the airworthiness authority throughout the MLDL to assist the certification process.
b. Gain agreement on the means of compliance through approval of the Plan for ML Aspects of Certification.
c. Provide compliance substantiation. The certification liaison process is applied as defined by the ML planning process (see Section 4) and the Plan for ML Aspects of Certification (see Section 11.1).

Section 10.2 Overview of MLIL Certification Liaison Process

The guidance provided by DO-178C Section 9.0 for certification liaison remains applicable to the MLIL.

Section 11 ML Lifecycle Output Data Items

ML output data items are produced during the ML Lifecycle to plan, direct, explain, define, record, and/or provide evidence of accomplishing processes, objectives, and activities. The output data items enable the ML Lifecycle processes, system, or equipment certification, and post-certification modification of the ML-based software item. The outputs also provide justified confidence for the accomplishment of the processes, objectives, and activities at the appropriate level of rigor, quality, and completeness.

The Annex A tables are a cross-reference that show how the output data items described in this section of the document are outputs of the ML lifecycle processes described in earlier sections of this document.

The ML lifecycle output data items are similar to the approach established in DO-178C Section 11.

The output data items should have the following attributes:

a. All output data items content should have the following attributes:
    1. Clear – The content of the output data items should be written to be clear and interpretable by the stakeholders of the artifacts.
    2. Complete – The content should not include gaps or incomplete areas but instead should be mature and complete.
    3. Consistent – The content style, and approach should be standardized across outputs.
    4. Managed – The content should be under appropriate configuration management.
    5. Accessible – The content should be appropriately accessible.
b. Output data items dealing with requirements, design descriptions, and verification should have the following attributes:
    1. Traceable – The content should include bi-direction traceability for requirements, design, and verification.
    2. Verifiable – The content should be able to be verified against requirements, design, or verification activities.

The guidance provided by DO-178C Section 11 Software Lifecycle Data for the creation of the output data items is applicable for the creation of the MLIL output data items. What follows is the description of the MLDL output data items.

Section 11.1 Plan For Machine Learning Aspects of Certification

The Plan for Machine Learning Aspects of Certification description is similar to the approach established in DO-178C Section 11.1.

Like the purposes of the Plan for Software Aspects of Certification (PSAC) for traditional software item development, the Plan for Machine Learning Aspects of Certification (PMLAC) captures the planned approach to accomplish the processes, objectives, and activities and produce the output data items that will enable the assurance to be established to receive certification of the ML-based item. This plan is provided to the airworthiness authority early to foster early and common understanding of the approach to ML lifecycle development assurance.

a. The Plan for Machine Learning Aspects of Certification (PMLAC) should identify the following:
    1. The system or subsystem into which the ML-based item is to be integrated,

2. The target hardware on which the ML-based item, when deployed, is to run,
3. The interfaces to other hardware and software items the ML-based item is to use, and
4. The safety requirements the ML-based item is to support or include, e.g., monitoring and mitigation of the ML-based item, where the ML-based item is to support being interrupted.

b. The Plan for Machine Learning Aspects of Certification (PMLAC) should describe the ML-based item functions, performance, type of ML, accuracy of the ML algorithm, timing, and recovery approaches.
c. The Plan for Machine Learning Aspects of Certification (PMLAC) should describe the operational design domain, edge/corner cases, and adverse cases to be supported by the ML-based item.
d. The Plan for Machine Learning Aspects of Certification (PMLAC) should describe the approach to addressing the ML concerns and recommendations.
e. The Plan for Machine Learning Aspects of Certification (PMLAC) should describe the MLDL, and the approaches used to establish ML data assurance and ML model assurance by following the processes, objectives, and activities discussed in this document.
f. The Plan for Machine Learning Aspects of Certification (PMLAC) should describe the dependency on external standards and guidelines.
g. The Plan for Machine Learning Aspects of Certification (PMLAC) should describe the MLDL internal transition criteria between processes and the transition criteria from the MLDL to the MLIL.
h. The Plan for Machine Learning Aspects of Certification (PMLAC) should describe any deficiencies, known or anticipated, in the planned approach to meet the processes, objectives, activities, and producing the output data item.
i. The Plan for Machine Learning Aspects of Certification (PMLAC) should describe the schedule for the accomplishment of the processes, objectives, activities, and output data items, and the plan for interaction with the certifying authority for the review of those artifacts.
j. The Plan for Machine Learning Aspects of Certification (PMLAC) should describe novel items that could require unique process approaches, e.g., the use of GPU computers or non-traditional software languages or tools.
k. The Plan for Machine Learning Aspects of Certification (PMLAC) should indicate any usage of third-party dependencies, e.g., sources of ML data sets or ML model architecture libraries.
l. The Plan for Machine Learning Aspects of Certification (PMLAC) should indicate the approach for tool qualification and validation, verification, and accreditation (VV&A) for tools used during the MLDL.

The details for the Plan for Software Aspects of Certification for the MLIL are covered in DO-178C Section 11.1.

Section 11.2 Machine Learning Development Plan

The Machine Learning Development Plan content is similar to the approach established in DO-178C Section 11.2. The ML Development Plan is a description of the ML development procedures and ML lifecycle(s) to be used to satisfy the MLDL development process objectives. In addition, the ML Development Plan consists of the ML Data Development Plan and the ML Model Development.

This plan should include:

a. Standards: Identification of the ML Data Requirements Standard, the ML Model Requirements Standard, the ML Data Standard, and the ML Model Standard for the project.
b. ML data and model development and processing environment: This should include the environment, software languages, frameworks, and other tools to be used.
c. ML concerns and recommendations: Describe the approach to addressing the ML concerns and recommendations.
d. Requirements: Describe the approach for the development of the ML requirements, ML model requirements, and ML data requirements.
e. Retraining: If retraining is to be conducted, address the catastrophic forgetting ML concern, and the approach for ensuring appropriate regression testing is conducted.
f. Leverage industry ML development approaches, e.g., Desiderata[71], CPMAI[72], CRISP-DM[73], and TDSP[74], others[75].
g. ODD: Discussion of the development approach to ensure coverage of the ODD, corner cases, edge cases, and adverse cases.

Section 11.2.1 Machine Learning Data Development Plan

The ML data development plan is a description of the ML data development procedures and data development processes used to satisfy the ML data development process objectives.

The ML data development plan should address the following considerations:

a. The ML data development plan should have a thorough description of the approach taken for developing the ML data set.
b. The ML data development plan should identify the source of the data set.
c. The ML data development plan should identify the tools to be used for the development of the data set, and if tool qualification and verification, validation, and accreditation is needed for those tools.
d. The ML data development plan should indicate how tools will be verified as appropriate for the creation of the ML data set and used such that they do not introduce errors into the ML data set.
e. The ML data development plan should identify the processes and techniques for the development of the data set to be fully representative of the ML data requirements.
f. The ML data development plan should identify the processes and techniques for ensuring the data set is complete and representative of the ML-based item's operational design domain.

[71] Assuring the machine learning lifecycle: Desiderata, methods, and challenges, R. ASHMORE, R. CALINESCU, AND C. PATERSON, 1905.04223 (2019), https://arxiv.org/pdf/1905.04223.pdf.
[72] Cognilytica Cognitive Project Management for Artificial Intelligence (CPMAI), https://www.cognilytica.com/cpmai/.
[73] CRoss Industry Standard Process for Data Mining (CRISP-DM), https://www.datascience-pm.com/crisp-dm-2/.
[74] Microsoft Team Data Science Process (TDSP), https://docs.microsoft.com/en-us/azure/architecture/data-science-process/overview.
[75] Kaakai, F., Dmitriev, K., Adibhatla, S., Baskaya, E. et al., "Toward a Machine Learning Development Lifecycle for Product Certification and Approval in Aviation," SAE Int. J. Aerosp. 15(2):127-143, 2022, https://doi.org/10.4271/01-15-02-0009.

g. The ML data development plan should identify the process and techniques for ensuring the data set includes appropriate robustness conditions, e.g., corner cases, edge cases, and adverse conditions.
h. The ML data development plan should identify the processes and techniques for the creation of independent data sets such that data leakage and other data set bleed over does not occur.
i. The ML data development plan should indicate the approach to the development of the data set in conformance with the ML data standards, e.g., data quality requirements.
j. The ML data development plan should indicate conformance with the ML data requirements standards for the development of the ML data requirements, i.e., the standard used to create the ML data requirements.
k. The ML data development plan should indicate that the development of the data will address the concerns identified in the system safety assessment, e.g., mitigation for the concerns identified in the hazard assessment.
l. The ML data development plan should describe the approach for the production of the ML Data Processing Description, which will fully capture the process to produce the data set used for training, validating, and testing the ML model.
m. The ML data development plan should identify the transition criteria necessary to transition from data set requirements development to data set development to data set verification and then to use of the data set for training, validation, and verification of the ML model.

Section 11.2.2 Machine Learning Model Development Plan

The ML model development plan is a description of the ML model development procedures and model development processes used to satisfy the ML model development process objectives.

This plan should include:

a. The ML model development plan should identify the ML Model Requirements Standards and ML Model Standards for the project.
b. The ML model development plan should provide a description of the ML model development processes.
c. The ML model development plan should indicate the approach to mitigating the ML concerns, e.g., safe exploration, reward shaping, value alignment, interruptibility, and distribution shift susceptibility.
d. The ML model development plan should indicate the transition criteria to be used to transition from ML model requirements development to ML model development to ML model training, validation and testing, and finally to ML model design document.
e. The ML model development plan should address the following:
    1. Design techniques. This should address: (i) type of model, e.g., neural network vs. decision tree, (ii) architecture of model, e.g., a single model vs. combination of sub-models, (ii) choice of hyperparameters, e.g., number of layers, number of neurons per layer, and type of activation function in each layer. Justification for selections that the appropriate ML model technique was selected should be provided.
    2. Coding techniques.
    3. Training techniques and activities, e.g., optimization algorithm, and model optimization such as compression and pruning, model optimizations, and stop criteria.

4. The model selection process, e.g., selecting simplicity vs. complexity, explainability vs. performance, size vs. speed, maturity vs. novelty, etc.
5. Optimizing generalization and avoiding underfitting and overfitting.
6. The model and training approach should be deterministic with respect to the performance repeatability.
7. The training strategy, e.g., ensemble training, curriculum learning, feature selection, cross validation, etc.

f. The ML model development plan should detail the ML model development environment by identifying the tools used for ML model design, development, training, and validation.

Section 11.3 Machine Learning Verification Plan

The Machine Learning Verification Plan is similar to the approach established in DO-178C Section 11.3. The ML Verification plan details the approach to conduct verification of the ML data set and the ML model for compliance with the ML requirement during the MLDL. The ML Verification plan should include the following:

a. The ML Verification plan should detail the organization responsible for conducting the verification of the ML data set and ML model and their proficiency with conducting such verification.
b. The ML Verification plan should indicate verification approach of the exit and entrance criteria between each phase within the MLDL.
c. The ML Verification plan should indicate the tools and environments used during the ML data set and ML model verification process.
d. The ML Verification plan should indicate coordination with configuration management to ensure the correct versions are used and clearly noted during the ML data set and ML model verification process.
e. The ML Verification plan should include indication of how deficiencies are determined and reported during the verification process.
f. The ML Verification plan should indicate the environment used to conduct the verification of the ML model, and the environment's representation of the target deployment hardware environment.
g. The ML Verification plan should detail the approach to ensure the ML concerns are verified and the ML recommendations are followed.
h. The ML Verification plan should indicate the approach for the coordination of the ML data sets, and then their use for the verification of the ML model.
i. The ML Verification plan should account for any ML data set and ML model reverification and ensure appropriate regression and performance verification occur.

In addition, the ML Verification Plan consists of two components: (1) ML Data Verification Plan, and (2) ML Model Verification Plan.

Section 11.3.1 Machine Learning Data Verification Plan

The ML Data Verification Plan is a description of the verification procedures to be used to satisfy the ML data verification process objectives. These procedures may vary by assurance level as defined by Annex A. The procedures will establish the approach to confirm that the MLDL data requirements, especially those for the operational design domain, are indeed fulfilled by the MLDL data sets.

The ML data verification plan should include:

a. The ML data verification plan should indicate the approach to ensure compliance of the ML data set with the ML data requirements, data hazard assessment, ML model development plan, and the ML data standard.
b. The ML data verification plan should indicate the approach to verify that the ML data set data set is representative of the operational design domain (ODD), and includes the appropriate edge cases, corner cases and adverse conditions in alignment with the ML requirements and ML data requirements.
c. The ML data verification plan should indicate the approach to ensure the independence of the data sets, e.g., the independence of the training, validation, and test data sets and the independence of the training and test simulations and scenarios.
d. The ML data verification plan should indicate the approach to verifying data set labeling accuracy.
e. The ML data verification plan should indicate the approach to verify the ML data set includes the appropriate attributes, features, sources, and signals.
f. The ML data verification plan should indicate the approach to verify the ML data set includes adequate scenarios representative of the ODD.
g. The ML data verification plan should indicate the ML data set sampling, cross-check, or other approach to be used to verify the data set.
h. The ML data verification plan should indicate the approach to verify the adequacy of the ML data set for the ML model.
i. The ML data verification plan should indicate the approach to verify the absence of simulation-to-real (sim-to-real) gaps between the ML data set and the ODD.
j. The ML data verification plan should indicate the approach for reporting ML data compliance with ML data requirements, ML data development plan, and ML data standards, e.g., explainability approach for ML data verification.
k. The ML data verification plan should indicate the approach to report ML data set deficiencies.
l. If COTS ML data set is used, the ML data verification plan should indicate the approach to verifying it is compliant with the previously mentioned concerns.

Section 11.3.2 Machine Learning Model Verification Plan

The ML Model Verification Plan is a description of the verification procedures to be used to satisfy the ML model verification process objectives. These procedures may vary by assurance level as defined by Annex A.

The ML Model Verification Plan is to provide verification procedures to confirm that the MLDL model requirements, for expected performance in the operational design domain as well as for robustness testing and the absence of unintended behaviors, are indeed fulfilled by the MLDL model.

The model verification plan should include:

a. The ML model verification plan should indicate the approach to verify the compliance of the ML model with the ML model requirements, safety assessment, ML model development plan, and the ML model standard.

b. The ML model verification plan should indicate the approach to document and report the completeness and representativeness of the ML data and the performance and explainability of the ML model.
c. The ML model verification plan should indicate the approach to ensure the ML model generalization during the ML model verification, e.g., ML data set, including scenarios, not previously seen by the ML model is used to ensure generalization.
d. The ML model verification plan should indicate how the trained ML model integrity should be evaluated from a safety, operational design domain context, system requirements, and system safety requirements context.
e. The ML model verification plan should indicate how to verify confidence with the use of COTS, GOTS, or opensource ML model framework (software), e.g., tool qualification of the opensource ML model framework.
f. The ML model verification plan should verify that the ML model fulfills the ML requirements for performance (accuracy, generalization, and stability) for nominal and robustness conditions.
g. The ML model verification plan should indicate how the ML data sets will be used to verify the performance, safety, and generalization of the ML model to meet the ML model requirements for performance, safety, and generalization.
h. The ML model verification plan should indicate how the ML data sets will be used to verify the robustness of the ML model against edge cases, corner cases, adverse cases, and outside-of-domain data sets and scenarios.The ML model verification plan should indicate the verification approach for complying with the ML recommendations, e.g., monitoring and mitigations.
i. The ML model verification plan should indicate the approach for verifying the ML concerns are mitigated, e.g., verify mitigation of value alignment and interruptibility.
j. The ML model verification plan should indicate the approach for reporting ML model compliance with ML model requirements, ML model development plan, and ML model standards, e.g., explainability approach for ML model verification.
k. The ML model verification plan should indicate the approach for verifying ML model generalization, e.g., ensure that the generalization requirements, including functional and non-functional aspects (e.g., corner cases identified for safety), are met. Within ML model generalization verification discussion include verification against performance assurance metrics, which include, but are not limited to, the rate of false positive and false negative predictions/classifications (confusion matrix), mean absolute error, mean square error, etc. For supervised learning setting, typical discussion of conducting performance assurance should involve verifying the ML model using the test data set.
l. The ML model verification plan should indicate the approach for verifying the ML model stability, i.e., verify small perturbations in the inputs do not trigger unintended or unacceptable behavior. The Machine Learning Model Verification Plan should indicate that the verification will address the expected level of perturbation that the ML model should sustain as defined in the ML model requirements. Different approaches can be used to demonstrate ML model stability, including analysis or tests.

Section 11.3.2.1 Machine Learning Model Verification Plan Criteria

The criteria for the ML model verification plan include the following: nominal testing and robustness testing. Nominal testing consists of testing the ML model performance (accuracy, generalization, and

stability) within the nominal operational design domain, up to edge cases. Robustness testing includes adverse stress testing, such as that used for the ACAS X program.

Section 11.4 MLDL Configuration Management Plans

The MLDL Configuration Management Plans description is similar to the approach established in DO-178C Section 11.4. That is the MLDL Configuration Management Plans should include the following:

a. The ML configuration management plans should establish configuration management processes for the MLDL.
b. The ML configuration management plans should include the ML data configuration management plan and the ML model configuration management plan.
c. The ML configuration management plans should include transition criteria, version control processes, change management processes, and backup/retrieval approaches for the ML data and the ML model.

In addition, the MLDL Configuration Management Plan should address the following two components: (1) ML Data Configuration Management Plan, and (2) ML Model Configuration Management Plan.

Section 11.4.1 MLDL Data Configuration Management Plan

The MLDL Data Configuration Management Plan should address the following:

a. The MLDL Data Configuration Management Plan should include the version and change management control of the ML data set from creation to usage through to program termination.
b. The MLDL Data Configuration Management Plan should include details for storage of the data set for preservation for stakeholder incident investigation purposes.
c. The MLDL Data Configuration Management Plan should include the versioning identification and data set integrity, which should detail the management of changes to the data set.
d. The MLDL Data Configuration Management Plan should include the integrity plan detailing the approach to allowing change control of the data set.
e. The MLDL Data Configuration Management Plan should detail the approach for the data set to keep independence among different divisions of the data set, e.g., preserve the integrity of independence of the training, validation, and test data sets.
f. The MLDL Data Configuration Management Plan should include descriptions of how changes to the data set are coordinated with the ML model development process to ensure an appropriate version of the data set is used for training, validation, and testing.
g. The MLDL Data Configuration Management Plan should appropriately cover video, samples, simulation environments, scenarios, and other supporting formats of data sets.
h. The MLDL Data Configuration Management Plan should detail the application of the configuration management processes through the ML lifecycle for the data set.
i. The MLDL Data Configuration Management Plan should detail that modifications and changes should only occur by appropriate personnel.

Section 11.4.2 Machine Learning Model Configuration Management Plan

The MLDL Model Configuration Management Plan should address the following:

a. The MLDL Model Configuration Management Plan should include the version and change management control of the ML model from creation to training, validation, and testing through to program termination.
b. The MLDL Model Configuration Management Plan should include details for storage of the ML model for preservation for stakeholder incident investigation purposes.
c. The MLDL Model Configuration Management Plan should include the versioning identification and ML model integrity, which should detail the controls for changes to the ML model.
d. The MLDL Model Configuration Management Plan should include the integrity plan detailing the allowing change control of the ML Model.
e. The MLDL Model Configuration Management Plan should detail the approach for the ML Model to preserve the integrity of the ML Model from creation to training, validation, and testing to MLIL transition.
f. The MLDL Model Configuration Management Plan should include descriptions of how changes to the ML Model are coordinated with the ML data set process to ensure an appropriate version of the data set is used for ML Model training, validation, and testing.
g. The MLDL Model Configuration Management Plan should appropriately cover neural networks as well as tabular solutions.

Section 11.5 MLDL Quality Assurance Plan

The MLDL Quality Assurance Plan description is similar to that established in DO-178C Section 11.5. That is the MLDL Quality Assurance Plan should include the following:

a. The MLDL Quality Assurance Plan should detail the quality evaluation program for the accomplishment of the processes, objectives, activities, and evaluation of the content within the output data items.
b. The MLDL Quality Assurance Plan should detail the transition criteria to be evaluated between the ML lifecycle processes, e.g., from the ML Planning Process to the ML Requirements Process.
c. The MLDL Quality Assurance Plan should detail the conformity reports to be produced to evaluate the transition criteria between ML lifecycle processes.
d. The MLDL Quality Assurance Plan should detail the milestone events to be evaluated during the ML lifecycle processes.
e. The MLDL Quality Assurance Plan should detail the sampling process for the review of processes, objectives, activities, and output data artifacts.
f. The MLDL Quality Assurance Plan should detail the review approach for the configuration management artifacts.
g. The MLDL Quality Assurance Plan should detail the traditional approach to the following items as well: independence of authority, timing of reviews, and records to be produced.

Section 11.6 Machine Learning Data Standard

ML Data Standards define the methods, rules, and tools to be used to develop the data sets. These standards should include:

a. The machine learning data standard should detail the development of a data set for a specific intended use, i.e., the data set serves a specific operational design domain (ODD) that will be detailed in the ML data requirements.

b. The machine learning data standard should indicate the approach for producing data sets that are independent. For example, the machine learning data standard should detail the need for at least three independent data sets: Training data, Validation data, and Test data, and indicate the approach for independent simulation environments and scenarios.
c. The machine learning data standard should indicate best practices for the development of data sets and usage of data sets, e.g., avoid data leakage, which is the test data set having knowledge of training data.
d. The machine learning data standard should detail the best practices for data collection and establishing the data set. These include:
    1. Data acquisition origin source: real world, synthetic, synthesized (augmented real), or mixed.
    2. Appropriate format and accuracy, precision, and resolution, e.g., for TSPI (Time, Space, Position Information) data set those terms could have the following meaning:
        - Format: CSV, JSON, or specialized formats like National Marine Electronics Association (NMEA) 0183 protocol.
        - Accuracy: The degree to which the measured position reflects the true position, often in meters.
        - Precision: The granularity of the recorded position, e.g., decimal degrees for latitude/longitude.
        - Resolution: The smallest detectable change in position, often in decimal degrees or meters.
        - Fidelity: This includes the degree of representation of the operational design domain, e.g., round earth model vs. ellipsoidal earth model.
    3. Appropriate bias: data should include appropriate diversity, e.g., geographical, seasonal, source, and parametric variation; however, deliberate over-sampling of underrepresented cases can have negative consequences on model performance.
    4. Sufficiency: enough volume of data, scenarios, and sources to train, validate, and test the ML model to the desired level of accuracy.
    5. Representativeness: captures the foreseeable environmental conditions of the intended operation, based on data attributes defined in the requirements set, and with regards to the capabilities of the target environment. Representativeness should ensure the data sets used for ML model training are appropriately “representative” of the operational design domain, systems requirements, systems safety requirements, and data set requirements.
e. The machine learning data standard should detail data processing guidance for the establishment of the data set, which should include the following:
    1. Sampling approach – simple random sampling, clustered sampling, stratified sampling, systematic sampling, multiphase sampling
    2. Guidance for processing
    3. Normalizing formats, units, etc.
    4. Binning data
    5. Data normalization
    6. Feature expansion for data handling (e.g., day or week, month of year, quarter, sequential days, date, etc.)

7. Data augmentation techniques (e.g., image rotation, flipping, cropping, synthetic data, surrogate sensor models)
8. Annotation process
9. Labelling
10. Feature engineering

f. The machine learning data standard should detail how the concerns with data are addressed, e.g., simulation-to-real (sim-to-real) concerns.
g. The machine learning data standard should detail the data quality requirements, e.g., accuracy, completeness, representativeness, timeliness, integrity, robustness.
h. The machine learning data standard should detail the quantification of uncertainty associated with the data set to be taken, i.e., approaches should be taken to quantify the variability and characteristics within the data set that may lead to unintended and unexpected behavior of the ML system.

Section 11.7 Machine Learning Model Standard

The ML Model Standard details the programming languages, methods, rules, and tools to be used to develop the model, and characteristics that should and should not be present in the model. These model characteristics should help ensure no erroneous or unanticipated behaviors are added to the model:

a. The ML model design standard should describe the design techniques and activities for the creation of the ML model.
b. The ML model design standard should describe the approach for determining the ML model architecture characteristics.
c. The ML model design standard should describe the approach to determining the number of layers, hidden layers, neurons, activation functions, and hyperparameters that should be used in the neural network, and whether the ML model should be a single or multiple neural networks.
d. The ML model design standard should describe how the ML model architecture identifies ML Model elements and describes how the different elements should be integrated to achieve the intended function(s).
e. The ML model design standard should describe the software coding techniques, approach, and activities, which should align with aviation industry practices for safe and airworthy development.
f. The ML model design standard should describe how ML model training techniques and activities will be accomplished, e.g., building hyper parameters, conducting model optimizations (pruning), establishing initial values, setting loss function, and determining metrics and acceptance criteria.
g. The ML model design standard should describe the triggers for re-training after certain optimization techniques are applied.
h. The ML model design standard should describe the approach for ML model robustness to erroneous, missing, and late signals, sources, features, and attributes.
i. The ML model design standard should describe the approach for addressing the ML model concerns, e.g., distribution shift, catastrophic forgetting, etc., and complying with the ML recommendations.

Section 11.8 Machine Learning Requirements Standards

ML Requirements Standards define the methods, rules, and tools to be used to develop the ML requirements, i.e., ML data requirements and ML model requirements. These standards should include:

a. The ML requirements standards should detail the methods used to develop ML requirements, such as structured methods, structured grammar, or standardize format and lay-out, e.g., use of Easy Approach to Requirements Syntax[76] or similar to create the ML requirements.
b. The ML requirements standards should detail notations to be used to express requirements, such as data flow diagrams and formal specification languages.
c. The ML requirements standards should detail the constraints on the use of the tools used for requirements development.
d. The ML requirements standards should detail the method to be used to provide derived requirements to the system processes.

In addition, the ML Requirements Standard is composed of the following two standards: (1) ML data requirement standard, and (2) ML model requirements standard.

Section 11.8.1 Machine Learning Data Requirements Standard

ML Data Requirements Standards define the methods, rules, and tools to be used to develop the ML data requirements. The standard should include:

a. The ML data requirements standard should indicate the intended use of the data set, e.g., for training, validation, and testing of SL models.
b. The ML data requirements standard should indicate the desired approach for sourcing data set, e.g., using verified and validated data source, qualified tool for the development or sampled from real data set, or synthetized from real data set.
c. The ML data requirements standard should indicate Data Quality Requirements (DQRs),
d. The ML data requirements standard should indicate the success metrics for the data set, e.g. accuracy, precision, and resolution.
e. The ML data requirement standard should indicate various elements of data format, e.g. features, attributes, and sources.
f. The ML data requirements standard should indicate the data correctness expectations and how to establish those expectations, e.g., for discrete data sets and simulation environments (sim-gyms).
g. The ML data requirements standard should indicate the approach for establishing data signals, and features.
h. The ML data requirements standard should indicate the approach for labelling the data set.
i. The ML data requirements standard should indicate the approach for establishing data set sufficiency.
j. The ML data requirements standard should indicate the approach for establishing scenarios which are complete and representative.

[76] Terzakis, John, "EARS: The Easy Approach to Requirements Syntax", Intel Corporation, Version 1.0 john.terzakis@intel.com, [URL: https://www.iaria.org/conferences2013/filesICCGI13/ICCGI_2013_Tutorial_Terzakis.pdf, Accessed: 10/16/2022]

k. The ML data requirements standard should indicate the approach for using or establishing the simulation environment.

Section 11.8.2 Machine Learning Model Requirements Standard

ML Model Requirements Standards define the methods, rules, and tools to be used to develop the ML model requirements. These standards should include:

a. The ML Model Requirements Standards should indicate the allowable amount of uncertainty versus determinism, e.g., stability (e.g., maximum tolerated perturbation on the inputs, maximum tolerated perturbation on the output(s)).
b. The ML Model Requirements Standards should indicate the type of Classification/Regression.
c. The ML Model Requirements Standards should indicate the expected generalization behavior inside the operational design domain.
d. The ML Model Requirements Standards should indicate the expected ML Model robustness behavior at the boundary, e.g., edge/corner case, of or outside the operational design domain e.g., specification of outliers or specification of adverse conditions,
e. The ML Model Requirements Standards should indicate the nonfunctional requirements, e.g., memory and throughput available to implement the model.
f. The ML Model Requirements Standards should indicate input and output properties.
g. The ML Model Requirements Standards should indicate ML Model generalization capability.
h. The ML Model Requirements Standards should indicate how the ML recommendations are addressed.

Section 11.9 Machine Learning Requirements

The Machine Learning Requirements description is similar to the approach described in DO-178C Section 11.9. In general, the ML Requirements should include the following:

a. The ML requirements should be based on the ML-based requirements allocated to the ML-based item from the systems process.
b. The ML requirements should address all the ML-based requirements allocated by the systems process.
c. The ML requirements should also account for the safety concerns allocated by the systems process, e.g., the safety concerns in the Data Hazard Assessment.
d. The ML requirements should address the performance, timings, and exceptions indicated in the ML-based requirements allocated to the ML-based item from the systems process.
e. The ML requirements should align with the interface expectations, i.e., input and output, from the systems process.
f. The ML requirements should account for any hardware, partitioning, software, or other dependencies indicated by the requirements allocated by the system process.
g. The ML requirements should account for any monitoring and mitigation strategies indicated by the requirements allocated by the system process.
h. The ML requirements should address all ML recommendations to eliminate ML concerns.

In addition, the Machine Learning Requirements are composed of, at least, the following two requirement components: (1) Machine Learning data requirements, and (2) Machine Learning model requirements.

Section 11.9.1 Machine Learning Data Set Requirements

Machine Learning Data Requirement is a definition of the ML data set requirements including the derived ML data set requirements. This data should include:

a. The ML data set requirements should include the operational design domain (ODD), which includes edge cases, corner cases, and in/out of domain cases.
b. The ML data set requirements should include data set distribution definition.
c. The ML data set requirements should include data set completeness, accuracy, and representativeness for the operational design domain.
d. The ML data set requirements should include data set labels.
e. The ML data set requirements should include the threshold for data set sufficiency.
f. The ML data set requirements should include expected data set metrics.
g. The ML data set requirements should include data set uniformity, homogeneity, and correlation expectations.
h. The ML data set requirements should include data set constraints.
i. The ML data set requirements should include data sets to be used for robustness, e.g., anomaly detection.
j. The ML data set requirements should include scenarios definitions to be used with the data set and the simulation environment.
k. The ML data set requirements should include environment definition for the ODD.
l. The ML data set requirements should include definition of independent constraints in the scenarios.
m. The ML data set requirements should include time dependencies for the scenarios used for the data set.

Section 11.9.2 Machine Learning Model Requirements

Machine Learning Model Requirement is a definition of the ML model requirements including the derived ML model requirements. The ML model requirements should include:

a. The ML model requirements should indicate the ML model inputs, e.g., characteristics of the ML model input sources, signals, rates, and format input to the ML sub-system in the target environment (interface requirements),
b. The ML model requirements should indicate the ML mode output, e.g., characteristics of the ML model output signals and rates and format from the ML sub-system in the target environment (interface requirements),
c. The ML model requirements should indicate what operational design domain should be supported.
d. The ML model requirements should indicate the robustness criteria, e.g., handling of edge/corner case inputs.
e. The ML model requirements should indicate the stopping criteria, e.g., the interruptible criteria, which is how the ML model handles being interrupted.

f. The ML model requirements should indicate the performance criteria, e.g., temporal dependencies, confusion matrix, success/failure rate, acceptable loss function, allowable uncertainty, etc.
g. The ML model requirements should indicate the bias and variance.
h. The ML model requirements should indicate the ML recommendations to be addressed to address the ML concerns.

Section 11.10 Machine Learning Data Processing Description

Machine Learning Data Processing Description defines the processing used to produce the ML data set. This should include a retrospective description of the data set processing activities that occurred during the data processing process. Details in the ML Data Processing Description should guarantee reproducibility of the data set, scenarios, and/or simulation environment used during the MLDL model training, validation, and verification processes, e.g., should the need arise for incidence or mishap investigation, the data set could be reproduced from the original inputs. At a minimum the ML Data Processing Description should include:

a. The ML Data Processing Description should include the approach used for cleaning, preparing, and processing the data, e.g., normalization, fixing up formats, binning data, correcting erroneous values, computation of features, etc.
b. The ML Data Processing Description should include any deviations from the ML data standard, ML data requirements standard, and ML data development standard.
c. The ML Data Processing Description should include the resulting accuracy, resolution, and quality of the annotated data set.
d. The ML Data Processing Description should include the level of confidence that data set is representative and complete for the operational design domain.
e. The ML Data Processing Description should include details to produce the data in the same way the inference model receives, i.e., the model produced during the MLDL.
f. The ML Data Processing Description should include details for addressing corner cases and edge cases.
g. Where necessary, the ML Data Processing Description should include details for the construction of the simulation environment, and scenarios.

Section 11.11 Machine Learning Model Description

ML model description defines the ML model architecture and the design description that will satisfy the MLDL requirements. The ML Model Description includes sufficient documentation of the ML Model to facilitate meeting the MLDL requirements. At a minimum, the ML Model Description establishes the design characteristics of ML model, which can support the implementation in the MLIL. This data should include:

a. The ML model description should include the design of the ML model based on the MLDL requirements, specifically the ML model requirements.
b. The ML model description should include the appropriate data and control flow.
c. The ML model description should include the ML model architecture, e.g., including details of the hidden layers and activation functions, along with detailing the hyperparameters and parameters values in the appropriate analytical, algorithmic syntax, pooling, stride, padding, etc.

d. ML model description should include adequate detail to ensure replication of the ML model in the MLIL.
e. The ML model description should include the ML Model inputs (list, types, size, definition, units, accuracy, resolution, distribution).
f. The ML model description should include the ML Model outputs (list, types, size, definition, units, accuracy, resolution).
g. The ML model description should include descriptions of the ML model execution environment.
h. The ML model description should include the internal detail of the ML model, including the necessary traditional and ML algorithm details.
i. The ML model description should include the detail any and all pre-processing, post-processing, monitoring and other architectural details of the ML model to accomplish the ML-based item requirements in the MLDL and has gone through MLDL model verification.
j. The ML model description should include any optimization performed after training, e.g., removing inactive neurons, mixed precision.
k. The ML model description should reflect the trained, validated, and tested ML model.
l. The ML model description should include the details of the design for handling performance of the ML model at and beyond the operational design domain.
m. The ML model description should include any timing, memory, and deployment considerations.
n. The ML model description should address all ML recommendations to eliminate ML concerns.

Section 11.12 Machine Learning Data Sets

ML data sets, e.g., training, validation, and test data sets, are used to train, validate, and test the ML model and are a collection of data that satisfy the ML data requirements. This data should include:

a. Fulfillment of the ML data standards, and ML data requirements.
b. Independence amongst the sets of data.
c. Representativeness of the operational design domain.
d. Inclusion of appropriate nominal and robustness data.
e. Sufficient quantity to reach adequate ML model performance.
f. Processed to ensure uniformity, homogeneity, correlation, completeness, and quality.

Section 11.13 Machine Learning Environment Configuration Index

The Machine Learning Environment Configuration Index identifies the environment configuration of the ML-based software item product. The ML environment configuration index consists of the newly added MLDL environment configuration index item and MLIL environment configuration index items, which are covered by the Software Environment Configuration Index. Only those newly added MLDL Environment Configuration Index Items are listed below, while those related to MLIL and covered by the Software Configuration Index are covered under DO-178C Section 11.15.

The MLDL environment index is written to aid reproduction of the MLDL lifecycle environment for data set regeneration, model retraining, model reverification, or ML model description modification, and should:

a. Identify the MLDL environment data set processing tools and environment.

b. Identify the MLDL model tools and libraries used during the ML model development, e.g., tools used during ML model training, validation, and testing.
c. Identify the MLDL data set and model environment used to verify the data set and model, for example, the data set testing and analysis tools.
d. Identify MLDL qualified tools and their associated tool qualification data.

Section 11.14 Machine Learning Configuration Index

The Machine Learning Configuration Index identifies the configuration of the ML software item product. The ML Configuration consists of the newly added MLDL Configuration Index items and the MLIL Configuration Index items. Only those newly added MLDL Configuration Index Items are listed below, while those expected for the MLIL Configuration Index (Software Configuration Index) are covered under DO-178C Section 11.16.

Note: The configuration index can contain one output data item or a set (hierarchy) of output data items. The configuration index can contain the output data items listed below or it may reference another configuration item or other configuration identified output data item that specifies the individual output data items and their versions.

At a minimum the MLDL configuration index should identify:

a. Machine Learning Data Processing Description (see Section 11.10)
b. Machine Learning Data Sets (see Section 11.12).
c. ML Model Description (see Section 11.11)
d. Previously developed Machine Learning Data Sets, if used in training of the updated ML-based software item.
e. MLDL output data items.
f. Instructions for building the MLDL Machine Learning model and Parameter Data Item Files, if any, including, for example, instructions and data for compiling and linking; and the procedures used to recover the MDL model for regeneration, testing, or modification.
g. Reference to the MLDL Environment Configuration Index (see Section 11.13).

The MLDL configuration Index Items listed above could be combined with the set of MLIL configuration index items, which is equivalent to the Software Lifecycle Configuration Index, to form the complete ML Lifecycle Configuration Index.

Section 11.15 Machine Learning Verification Cases and Procedures

The Machine Learning verification cases and procedures description is similar to the approach described in DO-178C Section 11.4. Specifically, the MLDL verification cases and procedures should include the following:

a. The ML verification cases and procedures should be based on the ML Verification Plan.
b. The ML verification cases and procedures should have clear and concise steps and criteria.
c. The ML verification cases and procedures should be repeatable, so should include initialization and configuration criteria.
d. The ML verification cases and procedures should include robustness and performance cases and procedures for the ML data and ML model that are based on the MLDL requirements.

e. The ML verification cases and procedures should track the usage of the data sets, scenarios, and simulation environment and their configurations.
f. The ML verification cases and procedures, which are applied during the MLDL model verification process, should be based on the operational design domain scenarios.
g. The ML verification cases and procedures should include scenarios, which can be based on techniques such as Sobol sensitivity analysis, also known as Variance-based sensitivity analysis, regular lattice, Monte Carlo techniques (Hemmersly) or other techniques to establish the necessary cases to robustly evaluate the ML model.
h. The ML verification cases and procedures should use the Data Hazard Analysis, ML model requirements, ML architecture, and ML requirements as inputs in determining the ML verification cases and procedures.
i. The ML verification cases and procedures should prevent the ML model from learning from the test data. This involves establishing verification cases and procedures that properly reserve the correct portions (i.e., quantities) of the test data and ensure the robustness (i.e., completeness and quality) of the test data set.
j. Where necessary, the ML verification cases and procedures should include cases and procedures to address all ML recommendations to eliminate ML concerns.

Section 11.16 Machine Learning Verification Results

The Machine Learning verification results description is similar to the approach described in DO-178C Section 11.4. Specifically, the MLDL verification cases and procedures should include the following:

a. The ML verification results should document the resulting ML data set and ML model robustness and nominal performance for the ML data and model in accordance with the MLDL requirements.
b. The ML verification results should provide explanation of the ML data and ML model performance with respect to the MLDL requirements, especially when behavior does not align with ML requirements.
c. The ML verification results should clearly communicate the verification results, e.g., generalization, stability and assessment of performance using appropriate metrics such as accuracy, percentage of false positives and false negatives, confusion matrix, F-score, etc.
d. The ML verification results should document how the MLDL requirements are covered.
e. The ML verification results should document and illustrate how the ODD, edge cases, and corner cases are covered.
f. The ML verification results should comply with the ML Verification Plan.
g. The ML verification results should document the results for each of the ML verification cases and procedures, and whether the criteria were met or not, where failures are appropriately adjudicated.
h. The ML verification results should document the bi-directional traceability or point to the trace data.
i. The ML verification results should document that all ML recommendations were appropriately addressed to eliminate ML concerns.

Section 11.17 Machine Learning Problem Reports

See DO-178C Section 11.17 Problem Reports.

Section 11.18 Machine Learning Inference Model

The Machine Learning Inference Model is a combination of the ML model elements and the necessary traditional software elements and is an output of the MLIL. An intermediate version might be provided as an output of the MLDL to accompany the ML model description. The Machine Learning Inference Model will be equivalent to the Executable Object Code discussed in DO-178C Section 11.12.

Section 11.19 Machine Learning Configuration Management Records

The Machine Learning Configuration Management Records should consist of adequate artifacts to substantiate the execution of the configuration management and change control process for the MLDL. That MLDL output data items that should, at a minimum, be controlled during this process should consist of the MLDL planning and standards artifacts, ML requirements, ML data and ML Data Processing Description, and the ML model and ML model description. Any changes to these should be appropriately tracked and under approval and configuration management level of change. Changes to those items, where appropriate, should be tracked through change review processes, and any changes should be evaluated for their impact. Change impact analysis may be necessary for certain changes to properly estimated and track the changes. Artifacts should exist to substantiate this process. The Machine Learning Configuration Management Records can be combined with the MLIL Configuration Management Records. However, the Machine Learning Configuration Management Records should allow the MLDL and the MLIL to be separately evaluated.

Section 11.20 Machine Learning Quality Assurance Records

The Machine Learning Quality Assurance Records should cover the programs conformance with the program's quality program. These records should encompass the quality audits with all aspects of the MLDL, which span from the MLDL planning process to the MLDL verification process, and transition to the MLIL process. The quality assurance records should confirm the program's compliance with the program plans to meet the necessary assurance objectives throughout these processes. They should capture completion and compliance with the intent for the production of ML data sets and ML models for flight critical applications. At a minimum Conformity Reports should be produced, but they may in addition take the form of audit sheets to keep appropriate records of completion, and the quality assurance personnel should be appropriately involved in all milestones and events to observe completion when the project is being executed. "After the fact" quality assurance is not effective for traditional software development nor for ML-based software development.

Section 11.21 Machine Learning Accomplishment Summary

The guidance from DO-178C Section 11.20 for the contents of the Software Accomplishment Summary remains appropriate for the guidance for the contents of the Machine Learning Accomplishment Summary. Moreover, the Machine Learning Accomplishment Summary should address any deviations from the Plan for Machine Learning Aspects of Certification, or the other plans and standards. The Machine Learning Accomplishment Summary should also address the following:

a. Provide confirmation that the appropriate recommendations were followed.
b. Identification of any open problem reports, requirements, or performance deviations, and their severity and consequence.
c. Identification of Machine Learning Configuration Index, which is to be transitioned to the MLIL.

d. Identification of the transition criteria established from the MLDL to the MLIL being met, and any deficiencies that remain and their impact.
e. Any ML model performance constraints or limitations should be identified.
f. Any ODD coverage constraints or limitations should be identified.

The Machine Learning Accomplishment Summary can be combined with the Software Accomplishments Summary. However, the elements of the Machine Learning Accomplishment Summary must be clearly identified with the Software Accomplishments Summary.

Section 11.22 Machine Learning Trace Data

The trace involves bi-directional traceability between the parent and the child, so should be able to be followed in both directions, i.e., from parent to child, and from child to parent.

For the ML Certification Criteria, at a minimum the traced data should involve the following:

a. Bi-directional trace from the systems requirements to the ML-based item requirements,
b. Bi-directional trace from the ML-based item to the ML data requirements and the ML Model requirements,
c. Bi-directional trace from the ML data requirements to the ML data set and data processing description,
d. Bi-directional trace from the ML model requirements to the ML model and ML model description,
e. Bi-directional trace from the ML model to the ML data set MLDL test cases and procedures.

The MLIL should establish the bi-directional traceability from the MLIL requirements to the MLDL requirements, and appropriate MLDL design artifacts, i.e., ML model description and ML Data Processing Description.

Section 12 Additional Considerations

This section covers additional topics which may require consideration during the planning phase as they may impact the assurance necessary for the ML-based item. That is, the choice of using some of these technologies may increase or decrease the necessary assurances required by the certifying authority. At their face they should not be used without coordination with the certifying authority to further explore the impact of these considerations on the assurances necessary for the ML-based item.

Section 12.1 Use of Previously Developed Software

In addition to guidance provided in Section 2.5.7 ML Reuse – Models and Data, for ML-based software items see SAE AIR 6988 Section 6.2.2.5 Licensing discussion.

Section 12.2 Tool Qualification Process

In general, the current guidance provided by DO-178C Section 12.2 Tool Qualification Process considerations, as well as DO-330 Software Tool Qualification Considerations, remain applicable for the ML lifecycle.

There is no guidance for the qualification of ML-tools, so coordinate with the certifying authority if those are being considered.

For tools that produce the synthetic data set used to train the supervised learning model, in addition to tool qualification, the use of verification, validation, and accreditation (VV&A) of the tool should also be considered. VV&A should be able to help further address the simulation-to-real (sim-to-real) concerns associated with synthetic data sets. For simulation environments, the same recommendation is also applicable, i.e., tool qualification and VV&A for simulation environments.

Section 12.3 Alternative Methods

In general, the current guidance provided by DO-178C Section 12.3 Alternative Methods considerations remain applicable for the ML lifecycle. Additions to that guidance that are applicable to the ML lifecycle are mentioned below.

Section 12.3.1 Exhaustive Testing

For complex ML models, exhaustive testing may not be possible on today's computing systems and simulation environments. Selective Sobol sensitivity analysis, also known as variance-based sensitivity analysis, may be possible on a statistically significant portion of the operational design domain. The combination of exhaustive testing with development assurance might be a viable path for some programs to ensure no unexpected behaviors are present within the ML model.

For low-dimensional cases (small number of inputs), grid-based or Monte Carlo methods could be used to perform exhaustive testing involving full coverage of the operational design domain. However, those approaches may not be appropriate for high-dimensional cases (large number of inputs).

Section 12.3.2 Multiple Versions of Dissimilar Software

SCSC-153 indicates the following objective for the ML based system: ARC1-5 "Incorrect computation outputs are tolerated". As shown in Figure 18, SCSC-153 provides the following recommendations for potentially establishing dissimilar ML architecture to handle incorrect outputs:

- 1st: "validity checker, developed using traditional software techniques" "detects an incorrect output then the computation is re-run, either using the same input (if the ML model includes a random element) or a suitably adjusted input".
- 2nd: "inspired by multiplex avionics systems" "uses a number of distinct computations (or ML models) in parallel, combining their output with some form of voting or aggregation function".
- 3rd: "features two channels, one of which uses AI (implemented as an ML model), the other of which is implemented using traditional software" "traditional software channel implements an always safe, but low (perhaps very low) performance algorithm, whereas the ML model offers higher performance with a concomitant risk of incorrect (i.e., unsafe) outputs".

Where discrepancies exist amongst data sets used to train the multiple versions of software (or where out-of-distribution conditions are detected), those findings should be provided by to the Data Governance process. When providing those findings, investigation should be conducted to determine the acceptability of the discrepancies.

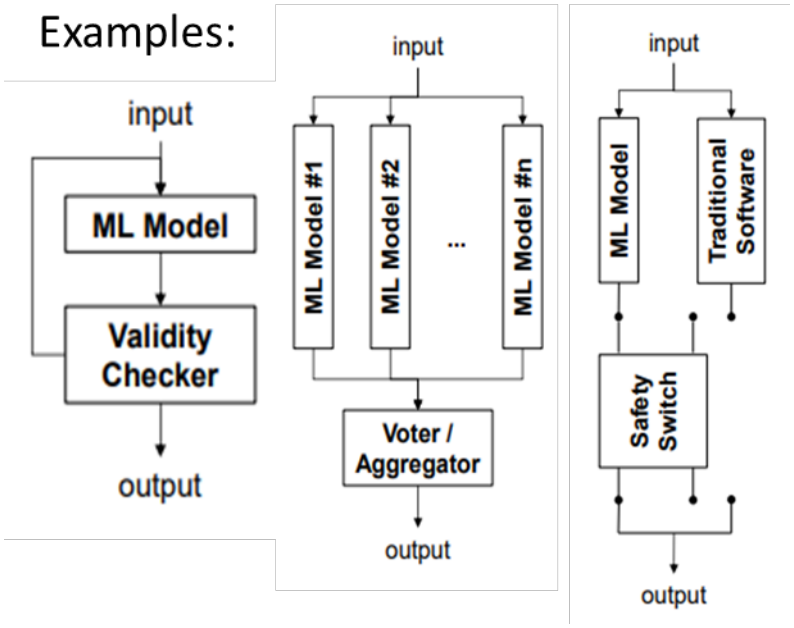

Figure 18. Example dissimilar ML architecture approaches. [Reference: SCSC-153]

In Figure 18, for example #2, the ML models, and for example #3, the traditional software, can be examples of multiple versions of dissimilar developed software. For such cases the guidance and activities provided by DO-178C Section 12.3.2 Considerations for Multiple-Version Dissimilar Software Verification would need to be followed should any of the objectives within this document be satisfied through the use of dissimilar software. Some of these topics are elaborated on by Ashmore et al[77] and worth reviewing.

[77] ASHMORE, R. CALINESCU, AND C. PATERSON, Assuring the machine learning lifecycle: Desiderata, methods, and challenges, arXiv, 1905.04223 (2019), https://arxiv.org/pdf/1905.04223.pdf.

Related to handling incorrect output is handling incorrect input, or input that is out of the operational design domain. Detection of out of domain can occur through the use of continuous monitors, and then an approach can be implemented of how to accommodate the detected out of domain input. The design should evaluate the value of considering and integrating such continuous monitors of inputs and outputs to handle incorrect inputs or outputs.

Section 12.3.3 Reliability Models

As with the guidance for using reliability models for traditional software development assurance, the same uncertainty remains for ML. Thus, for ML-based software items reliability models for ML-based items cannot be recommended for consideration.

Section 12.3.4 Product Service History

Software of unknown pedigree (SOUP) is not acceptable for traditional software items, nor for ML-based software items. For ML-based software items see SAE AIR 6988 Section 6.2.2.6 In-Service Experience for the expectations and for traditional software see DO-178C Section 12.3.4 Product Service History.

Section 12.4 Fielding Considerations

In general fielding considerations are beyond the scope of this document. However, some fielding concerns that should be factored into the ML lifecycle are provided below.

Section 12.4.1 Human Interaction and Training

In operation, the system user should sufficiently understand the ML behavior to use it as intended. AMACC Section 9 Human Systems Integration provides an abundance of guidance and considerations for properly evaluating the interaction of the human and the system. That guidance should be followed for evaluating and addressing the considerations associated with the human and Machine Learning interaction. Moreover, SCSC-153 provides the following best practices for accounting for those considerations:

- ARC2-1: Relevant information is presented to interacting parties. (Architecture/Information)
- ARC2-2: Relevant information is available to support maintenance and future development.
- PLT3-1: Safety-related demands on people interacting with the platform are reasonable.
- PLT3-2: Suitable interfaces are provided for people that may interact with the platform. (People)
- PLT3-3: Appropriate training is provided for platform users and maintainers. (Platform/People)

Guidance provided in SCSC-153 for each of the above should be taken to heart when constructing ML-based systems that will be interacting with humans.

Section 12.4.2 Post Incident Investigation

After an in-service occurrence, the manufacturer should be able to reproduce what happened and explain the causes of the occurrence (large-capacity data recording could be necessary). That is, operational data feedback to support mishap investigation should be collected. This should include success and failure cases being captured, managed, and assessed. As indicated by SCSC-153B COM3-4 and ARC2-3, “sufficient information needs to be recorded to allow the algorithm’s behavior to be reconstructed after the incident ... storing internal state information, including any data used to support non-deterministic choices within the algorithm…demonstrate that post-incident analysis can be conducted…[for example] treating discoveries during development and testing as pseudo-incidents and

confirming that sufficient information was recorded to support post-incident analysis…Learning from experience is an important part of a mature safety culture."

Section 12.4.3 AI/ML Deployment Considerations

Along with unique and challenging development concerns, target hardware deployment concerns exist for artificial intelligence (AI) and machine learning (ML) applications. Those deployment concerns should be addressed in the planning phase and consist of the issues surrounding the target hardware selection and the certifiability/qualifiable of the target hardware for the AI/ML model deployment. These concerns center around certification issues identified for multi-core processors (MCP), where those MCP issues are amplified for graphics processor units (GPUs) when they are used for general computing. While the use of complex graphics processors for general computing is being reconciled for flight critical applications, the reduction of these concerns is possible through design specific target hardware choices, e.g., selection of Field Programmable Gate Array (FPGA) devices or other certifiable approaches. Treatment of this topic is beyond the scope of this document, but treatment of the topic is addressed in: "Navigating Airworthiness Concerns with Deploying AI/ML Applications – A Brief Survey"[78].

[78] Carter, G., Chan, A., Terres, V., Rupert, J., Scales, A., "Navigating Airworthiness Concerns with Deploying AI/ML Applications – A Brief Survey", Vertical Flight Society (VFS) Forum 80, Montreal, Canada, DOI: 10.4050/F-0080-2024-0032.

Annex A ML Objectives, Activities, and Output Data Item per Item Development Assurance Level

The Annex A tables primarily present the MLDL objectives, activities, and output data items. Since the MLIL is to be primarily covered by traditional software development lifecycle, e.g., DO-178C objectives, activities, and output data items, the MLIL objectives, activities, and output data items are only mentioned in the Annex A tables when necessary. Otherwise, the reader is referred to DO-178C Annex A for a full listing of the MLIL (i.e., software lifecycle) process objectives, activities, and output data items. The MLDL objectives, activities, and output data items covered below combine with the MLIL objectives, activities, and output data items, which are covered in DO-178C Annex A, to form the full listing of ML lifecycle objectives, activities, and output data items.

As with DO-178C Annex A tables, the following process objective, activities, and output data item tables should not be used as a checklist without reference to the rest of the document. These tables do not reflect all complex aspects of compliance covered in this document. To fully understand the guidance, the full body of this document should be considered.

References (Ref) in these tables point to those sections in the text that define the objectives, related activities, and output data items. The tables include guidance for:

a. The process objectives are applicable for each development assurance level for levels A through DThe independence by development assurance level of the ML lifecycle process activities is applicable to satisfy that process's objectives.
b. The control category by development assurance level for the ML lifecycle output data item produced by the ML lifecycle process activities (see Section 8.1.3).

The following legend applies to "Applicability by Development Assurance Level (DAL)" and "Control Category (CC) by Development Assurance Level (DAL)" for all tables:

Legend:	●	The objective should be satisfied with independence.
	❍	The objective should be satisfied.
	Blank	Satisfaction of objective is at the applicant's discretion.
	①	Data satisfies the objectives of Control Category 1 (CC1).
	②	Data satisfies the objectives of Control Category 2 (CC2).

Annex A-1 ML Lifecycle Planning Process

	Objective		Activity	Applicable DAL				Output[2]		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	The activities of the ML lifecycle processes are defined.[1]	4.1.a	4.2 4.3.1	❍	❍	❍	❍	Plan For ML Aspects of Certification ML Data & Model Development Plan ML Data & Model Verification Plan ML Data & Model Configuration Management Plan ML Data & Model Quality Assurance Plan	11.1 11.2 11.3 11.4 11.5	①	①	①	①
2	The ML lifecycle is defined, including the inter-relationships between processes, sequencing, feedback mechanisms, and transition criteria.[1]	4.1.b	4.2 4.3.1	❍	❍	❍		Plan For ML Aspects of Certification ML Data & Model Development Plan ML Data & Model Verification Plan ML Data & Model Configuration Management Plan ML Data & Model Quality Assurance Plan	11.1 11.2 11.3 11.4 11.5	①	①	①	
3	ML lifecycle environment is selected and defined.[1]	4.1.c	4.2.c 4.2.i 4.4	❍	❍	❍		Plan For ML Aspects of Certification ML Data & Model Development Plan ML Data & Model Verification Plan ML Data & Model Configuration Management Plan ML Data & Model Quality Assurance Plan	11.1 11.2 11.3 11.4 11.5	①	①	①	
4	Additional ML considerations are addressed.[1]	4.1.d	4.2	❍	❍	❍	❍	Plan For ML Aspects of Certification ML Data & Model Development Plan ML Data & Model Verification Plan ML Data & Model Configuration Management Plan ML Data & Model Quality Assurance Plan	11.1 11.2 11.3 11.4 11.5	①	①	①	①
5	ML lifecycle development standards are defined.[1]	4.1.e	4.2 4.5	❍	❍	❍		ML Data & Model Requirements Standard ML Data Standard ML Model Standard	11.8 11.6 11.7	②	②	②	
6	ML lifecycle plans comply with this document. [1]	4.1.f	4.3 4.6	❍	❍	❍		ML Verification Results	11.16	②	②	②	

	Objective		Activity	Applicable DAL				Output[2]		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
7	Development and revision of ML lifecycle plans are coordinated. [1]	4.1.g	4.3 4.6	❍	❍	❍		ML Verification Results	11.16	②	②	②	

Notes:

[1] ML lifecycle includes the MLDL and the MLIL. MLIL objectives are addressed in Software Lifecycle, i.e., DO-178C.

[2] Only the MLDL outputs are listed. MLIL outputs are addressed in Software Lifecycle, i.e., DO-178C.

Annex A-2 MLDL Lifecycle ML Requirements Process

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	ML Data requirements are developed.	5.1.1.1.a	5.1.1.2	❍	❍	❍	❍	ML Data Requirements ML Trace Data	11.9.1 11.22	①	①	①	①
2	Derived ML Data requirements are defined and provided to the system process, including the system safety assessment process.	5.1.1.1.b	5.1.1.2.h 5.1.1.2.i	❍	❍	❍		ML Data Requirements ML Trace Data	11.9.1 11.22	①	①	①	
3	ML Model requirements are developed.	5.1.2.1.a	5.1.2.2	❍	❍	❍		ML Model Requirements ML Trace Data	11.9.2 11.22	①	①	①	
4	Derived ML Model requirements are defined and provided to the system process, including the system safety assessment process.	5.1.2.1.b	5.1.2.2.h 5.1.2.2.i	❍	❍	❍		ML Model Requirements ML Trace Data	11.9.2 11.22	①	①	①	

Annex A-3 MLDL Data Governance Process

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	Data sets are developed in accordance with ML data requirements, ML data development plan and standards.	6.1.1.a	6.1.2	❍	❍	❍	❍	ML Data Processing Description ML Data Set	11.10 11.12	①	①	①	①
2	Appropriate independent data sets are developed for ML model development, e.g., training, validation, and test data sets.	6.1.1.b	6.1.2.g 6.1.2.h 6.1.2.l	❍	❍	❍		ML Data Processing Description ML Data Set	11.10 11.12	①	①	①	
3	The ML Data Processing Description is developed and provided for the verification and implementation processes.	6.1.1.c	6.1.2.m	❍	❍	❍		ML Data Processing Description	11.10	①	①	①	

Annex A-4 MLDL Model Development Process

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	Develop the ML model in accordance with ML requirements, ML development plans and standards.	6.2.1.a	6.2.2	●	❍	❍	❍	ML Model Description	11.11	①	①	①	①
2	Develop ML model architecture.	6.2.1.b	6.2.2.d 6.2.2.e	●	❍	❍		ML Model Description	11.11	①	①	①	
3	Train ML model on ML training data set.	6.2.1.c	6.2.2.f 6.2.2.g	●	❍	❍	❍	ML Model Description	11.11	①	①	①	①
4	Validate the ML model hyperparameters by way of the ML validation data set.	6.2.1.d	6.2.2.h	●	❍	❍		ML Model Description	11.11	①	①	①	
5	ML model description is developed	6.2.1.e	6.2.2.j	●	❍	❍		ML Model Description	11.11	②	②	②	

.

Annex A-5 Verification of Outputs from the MLDL Requirements Process

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	ML requirements[1] comply with system requirements.	6.3.3.1.a	6.3.3.1	●	❍	❍	❍	ML Verification Results ML Trace Data	11.16 11.22	①	①	①	①
2	ML requirements[1] are accurate and consistent.	6.3.3.1.b	6.3.3.1	●	❍	❍		ML Verification Results	11.16	①	①	①	
3	ML requirements[1] are compatible with the operational design domain.	6.3.3.1.c	6.3.3.1	●	❍	❍		ML Verification Results	11.16	①	①	①	
4	ML requirements[1] are verifiable.	6.3.3.1.d	6.3.3.1	●	❍	❍		ML Verification Results	11.16	①	①	①	
5	ML requirements[1] conform to standards.	6.3.3.1.e	6.3.3.1	●	❍	❍	❍	ML Verification Results	11.16	②	②	②	②
6	ML requirements[1] are traceable to system requirements.	6.3.3.1.f	6.3.3.1	●	❍	❍		ML Verification Results ML Trace Data	11.16 11.22	②	②	②	
7	ML algorithms are accurate.	6.3.3.1.g	6.3.3.1	●	❍	❍		ML Verification Results	11.16	①	①	①	

Notes:

[1] ML requirements include ML data set requirements and ML model requirements.

Annex A-6 Verification of Outputs from the MLDL Data Governance Process

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	**Description**	**Ref**	**Ref**	**A**	**B**	**C**	**D**	**Data Item**	**Ref**	**A**	**B**	**C**	**D**
1	ML data sets comply with ML data requirements and data hazard analysis.	6.3.3.2.a	6.3.3.2 6.3.5	●	❍	❍	❍	Machine Learning Verification Cases and Procedures ML Verification Results ML Trace Data	11.15 11.16 11.22	①	①	①	①
2	ML data sets comply with the operational design domain.	6.3.3.2.b	6.3.3.2	●	❍	❍		ML Verification Results	11.16	①	①	①	
3	ML data sets are verifiable.	6.3.3.2.c	6.3.3.2	●	❍	❍		ML Verification Results	11.16	①	①	①	
4	ML data sets conform to standards.	6.3.3.2.d	6.3.3.2	●	❍	❍		ML Verification Results	11.16	①	①	①	①
5	ML data sets are traceable to ML data requirements.	6.3.3.2.e	6.3.3.2 6.3.5	●	❍	❍	❍	Machine Learning Verification Cases and Procedures ML Verification Results ML Trace Data	11.15 11.16 11.22	②	②	②	
6	ML data sets are accurate and consistent.	6.3.3.2.f	6.3.3.2	●	❍	❍		ML Verification Results	11.16	②	②	②	
7	ML data sets are adequate and sufficient.	6.3.3.2.g	6.3.3.2	●	❍	❍		ML Verification Results	11.16	①	①	①	①

Annex A-7 Verification of Outputs from the MLDL ML Model Architecture Process

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	ML model architecture complies with ML requirements.	6.3.3.3.a	6.3.3.3 6.3.5	●	❍	❍	❍	ML Verification Results	11.16	①	①	①	①
2	ML model architecture is accurate and consistent and complies with the ML model design.	6.3.3.3.b	6.3.3.3	●	❍	❍		ML Verification Results	11.16	①	①	①	
3	ML model architecture is verifiable.	6.3.3.3.c	6.3.3.3	●	❍	❍		ML Verification Results	11.16	①	①	①	
4	ML model architecture complies with the ML standards.	6.3.3.3.d	6.3.3.3	●	❍	❍		ML Verification Results	11.16	①	①	①	
5	ML model architecture demonstrates bi-directional traceability to ML model requirements.	6.3.3.3.e	6.3.3.3	●	❍	❍	❍	ML Verification Results	11.16	②	②	②	①
6	ML model architecture is robust to adverse cases, edge cases, and corner cases.	6.3.3.3.f	6.3.3.3	●	❍	❍	❍	ML Verification Results	11.16	②	②	②	①

Annex A-8 Verification of Outputs from the MLDL ML Model Development Process

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	ML model complies with ML model requirements.	6.3.3.4.a	6.3.3.4 6.3.5	●	❍	❍	❍	Machine Learning Verification Cases and Procedures ML Verification Results ML Trace Data	11.15 11.16 11.22	①	①	①	①
2	ML model complies with the ML model architecture.	6.3.3.4.b	6.3.3.4	●	❍	❍		ML Verification Results	11.16	①	①	①	
3	ML model is verifiable.	6.3.3.4.c	6.3.3.4	●	❍	❍		ML Verification Results	11.16	①	①	①	
4	ML model conforms to standards.	6.3.3.4.d	6.3.3.4	●	❍	❍		ML Verification Results	11.16	①	①	①	
5	ML model is traceable to ML model requirements.	6.3.3.4.e	6.3.3.4 6.3.5	●	❍	❍	❍	Machine Learning Verification Cases and Procedures ML Verification Results ML Trace Data	11.15 11.16 11.22	②	②	②	①
6	ML model is accurate and consistent.	6.3.3.4.f	6.3.3.4	●	❍	❍		ML Verification Results	11.16	②	②	②	

Annex A-9 MLDL Verification of Verification Process

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	**Description**	**Ref**	**Ref**	**A**	**B**	**C**	**D**	**Data Item**	**Ref**	**A**	**B**	**C**	**D**
1	MLDL test cases and procedures are correct.	6.3.4.5.a 6.3.4.5.b	6.3.4.5	●	❍	❍	❍	Machine Learning Verification Cases and Procedures ML Verification Results	11.15 11.16	①	①	①	①
2	MLDL test results are correct, and performance, behavior, and explanation, i.e., justification and rationale, are provided when behavior does not align with ML requirements.	6.3.4.5.c 6.3.4.5.d 6.3.4.5.e	6.3.4.5	●	❍	❍		ML Verification Results	11.16	①	①	①	
3	MLDL test coverage of ML requirements[1] has been achieved.	6.3.4.4.a	6.3.4.4.1	●	❍	❍		ML Verification Results ML Trace Data	11.16 11.22	①	①	①	
4	ML test coverage of ML data and model structure is achieved.	6.3.4.4.b	6.3.4.4.2 6.3.4.4.3	●	❍	❍		ML Verification Results	11.16	①	①	①	
5	ML test coverage of ML data and model nominal and robustness performance is achieved.	6.3.4.4.c	6.3.4.2 6.3.4.2.1 6.3.4.2.2	●	❍	❍		ML Verification Results	11.16	①	①	①	

Notes: [1] ML requirements include ML data set requirements and ML model requirements.

Annex A-10 MLIL Development Process

Only MLIL development objectives, activities, and output data items unique to ML that add to the existing Software Development Process objectives, activities, and output are listed below. Reference DO-178C Table A-2 for the existing Software Development Process objectives, activities, and output data items. Moreover, all Software Lifecycle processes, objectives, and activities proceeding and following the MLIL Development Process should also be accomplished. Those are listed in DO-178C Annex A.

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	MLIL inference model complies with the ML requirements and descriptions, i.e., exact replication.	7.1.a	7.1	❍	❍	❍	❍	ML Inference Model	11.18	①	①	①	①
2	MLIL inference model deviations, e.g., optimizations, from the ML Model requirements and description are documented, i.e., justified.	7.1.b	7.1	❍	❍	❍	❍	ML requirements and model description (update)	11.11	①	①	①	①

Annex A-11 MLIL Verification[1] Process

Only MLIL verification process objectives, activities, and output data items added to the existing Software Lifecycle Verification Process objectives, activities, and data items are listed below. All Software Lifecycle Verification Process objectives, activities, and outputs apply to the MLIL Verification Process, i.e., DO-178C Table A-4, A-5, A-6, and A-7.

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	MLIL Implementation transition criteria are achieved.	7.2.a	7.2	❍	❍	❍		MLIL Verification Results	DO-178C 11.14	②	②	②	
2	MLIL inference model verified against the ML requirements.	7.2.b	7.2	●	❍	❍	❍	MLIL Verification Results Trace Data	DO-178C 11.14 & DO-178C 11.21	①	①	①	①
3	MLIL model deviations from the ML requirements and ML Model Description are verified.	7.2.c	7.2	●	❍	❍		MLIL Verification Results Trace Data	DO-178C 11.14 & DO-178C 11.21	①	①	①	

Notes:

[1] Additional MLIL verification objectives and activities may be necessary to replace the loss in meaning of structural coverage analysis.

Annex A-12 ML Lifecycle Configuration Management Process

Configuration management objectives, activities, and output data items apply to MLDL and MLIL.	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	**Description**	**Ref**	**Ref**	**A**	**B**	**C**	**D**	**Data Item**	**Ref**	**A**	**B**	**C**	**D**
1	MLDL configuration items are identified.	8.1.1.a	8.1.2.1 8.1.3	❍	❍	❍	❍	ML Configuration Management Records	11.19	②	②	②	②
2	MLDL traceability and baseline is established, e.g., for data set, model, data processing description and ML model description.	8.1.1.b	8.1.2.2 8.1.3	❍	❍	❍	❍	MLDL Configuration Index ML Configuration Management Record	11.14 11.19	① ②	① ②	① ②	① ②
3	MLDL problem reports, change control, change reporting, and change status management are established.	8.1.1.c 8.1.1.d 8.1.1.e 8.1.1.f	8.1.2.3 8.1.2.4 8.1.2.5 8.1.2.6 8.1.2.7 8.1.3	❍	❍	❍	❍	Machine Learning Problem Reports ML Configuration Management Record	11.17 11.19	②	②	②	②
4	4Archive, retrieval, and release process established.	8.1.1.g	8.1.2.4 8.1.2.5 8.1.2.6 8.1.2.7 8.1.3	❍	❍	❍	❍	ML Configuration Management Record	11.19	②	②	②	②

Configuration management objectives, activities, and output data items apply to MLDL and MLIL.	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	**Description**	**Ref**	**Ref**	**A**	**B**	**C**	**D**	**Data Item**	**Ref**	**A**	**B**	**C**	**D**
5	MLDL environment control is established, and the environment is both archived and retrievable.	8.1.1.h	8.1.5	❍	❍	❍	❍	ML Environment Configuration Index ML Configuration Management Record	11.13 11.19	① ②	① ②	① ②	① ②

Annex A-13 ML Lifecycle Quality Assurance Process

Quality assurance objectives, activities, and output data items apply to MLDL and MLIL. Special attention to the MLDL quality assurance objectives, activities, and output data items in the table below. Refer to Section 9.1 for the remainder, and for the MLIL quality assurance objectives, activities, and output data items refer to RTCA DO-178C Table A-9.

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	MLDL plans, standards, process, objectives, activities, and output data items are developed appropriately.	9.1.1.a	9.1.2	●	●	●	●	Machine Learning Quality Assurance Records	11.20	②	②	②	②
2	Assurance is obtained that MLDL processes comply with plans and standards.	9.1.1.b	9.1.2	●	●	●	●	Machine Learning Quality Assurance Records	11.20	②	②	②	②
3	Assurance is obtained that transition criteria for the MLDL processes are satisfied.	9.1.1.c	9.1.2	●	●	●	●	Machine Learning Quality Assurance Records	11.20	②	②	②	②
4	Assurance is obtained that the MLDL artifact conformity review is conducted.	9.1.1.d	9.1.2	●	●	●	●	Machine Learning Quality Assurance Records	11.20	②	②	②	②

Annex A-14 ML Lifecycle Certification Liaison Process

All Software Lifecycle certification liaison objectives, activities, and output data items apply to MLDL and MLIL. MLDL certification liaison assurance objectives, activities, and output data items are provided below, and for MLIL assurance objectives, activities, and output data items refer to RTCA DO-178C Table A-10.

	Objective		Activity	Applicable DAL				Output		CC by DAL			
ID	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	An assurance plan is established between the applicant and the airworthiness authority that an ML Lifecycle will be followed.	10.1.a	10.1	❍	❍	❍	❍	Plan for ML Aspects of Certification	11.1	①	①	①	①
2	The method of compliance is proposed and agreement with the Plan for ML Aspects of Certification is obtained.	10.1.b	10.1	❍	❍	❍	❍	Plan for ML Aspects of Certification	11.1	①	①	①	①
3	Compliance substantiation is provided, especially for the transition from the MLDL to the MLIL.	10.1.c	10.1	❍	❍	❍	❍	ML Accomplishment Summary ML Configuration Index	11.21 11.14	①	①	①	①

Annex B Acronyms and Glossary of Terms

A full listing of acronyms used in this document are found in the following table.

Acronym	Definition
AC	Advisory Circular
AFE	Authorization for Expenditure (for AVSI Projects)
AFHA	Aircraft functional hazard assessment
AI	Artificial Intelligence
AMACC	Army Military Airworthiness Certification Criteria
AMCOM	Aviation and Missile Command
ANN	Artificial Neural Networks
ANSI	American National Standards Institute
AR	ARMY Regulation
ARINC	Aeronautical Radio, Incorporated
ARP	Aerospace Recommended Practice
ASA	Aircraft Safety Assessment
AvMC	Aviation and Missile Center (for Army DEVCOM)
AVSI	Aerospace Vehicle Systems Institute
CAST	Certification Authorities Software Team (for FAA)
CC	Control Category
CFG	Configuration file
CM	Configuration Management
CMP	Configuration Management Plan
COTS	Commercial off the shelf
CPU	Central processing unit
CSV	Comma separated values
CTA	Consumer Technology Association
CV-HAZOP	Computer-vision Hazard & Operability study
DAL	Development assurance level
DEVCOM	Army Combat Capabilities Development Command
EASA	European Union Aviation Safety Agency
FDAL	Functional Development Assurance Level
FHA	Functional hazard assessment
FMEA	Failure modes effects analysis
FMES	Failure modes effect summary
FTA	Fault tree assessment
GPU	Graphics Processing Unit
HLRs	High-level requirements

Acronym	Definition
IDAL	Item Development Assurance Level
IMA	Integrated modular avionics
LLRs	Low-level requirements
MBD	Model-based development
ML	Machine learning
MLDL	ML Development Lifecycle
MLIL	ML Implementation Lifecycle
NIST	National Institute of Standards and Technology
ODD	Operational design domain
PASA	Preliminary aircraft safety assessment
PHA	Preliminary Hazard Analysis
PSSA	Preliminary system safety assessment
RTOS	Real-time operating system
SAS	Software Accomplishments Summary
SCSC	Safety-Critical Systems Club
SFHA	System functional hazard assessment
SOUP	Software of unknown pedigree
SRD	Systems Readiness Directorate (for Army DEVCOM)
SSA	System safety assessment
STAMP	System-theoretic model of accidents
STPA	System-Theoretic Process Analysis
TPU	Tensor processing Unit
UAS	Unmanned aircraft system

A subset of necessary definitions used in this document are provided in the following table. A more extensive glossary of ML-related terms can be found in the AI-2363 Phase 1 paper as well as other sources[79]. The following terms apply to this document's scope and assurance approach. More exotic definitions exist for some of the following terms, but they were not useful for the purposes of this document. Instead, the useful portion of those definitions are maintained and presented here.

Term	Definition
Adaptive Learning	The inference model is allowed to update the weights of the ML model while fielded/deployed. Within this document, locked is synonymous with frozen and non-adaptive, while antonyms are online, continuous, unlocked, and adaptive.
Adaptively Collected Data Sets	Data sets collected in a manner influenced by the model's prediction or decisions.

[79] ANSI/ Consumer Technology Association (CTA) Standard, Definitions and Characteristics of Artificial Intelligence, ANSI/CTA-2089.

Term	Definition
Airworthy	1. The property of a particular air system configuration to safely attain, sustain, and terminate flight in accordance with the approved usage and limits. [MIL-HDBK-516C] 2. Airworthy means the aircraft conforms to its type design and is in a condition for safe operation. [FAA Far 3:5(a)]
Airworthiness Determination	"the process of assessing the capability of the aircraft system and/or subsystem to meet the approved airworthiness requirements throughout the system and/or subsystem lifecycle." [Reference: Army Regulation (AR) 70-62]
Artificial Intelligence	The use of ML and other techniques to automate the execution of functional behavior.
Assurance	The establishment of justified confidence through evidence that processes, objectives, and activities have been completed to the appropriate level of rigor.
Autonomy	"The extent to which a system [, through the aggregated usage of AI and other techniques,] can carry out its own [independent] processes and operations without external control." [Beers, Fisk and Rogers, 2014] [National Aeronautics and Space Administration[80]].
Certification basis	"The complete (necessary and sufficient), documented set of airworthiness criteria, standards and methods of compliance utilized to assess the airworthiness and safety of flight of a specific system design." [MIL-HDBK-516C 3.1.12]
Completeness (Data Set)	Training, validation, and test data sets should address all features, attributes, signals, and sources expected to be encountered during deployment in the operational design domain.
Component	An element of an item that accomplishes one or more functions.
Data Set	A collection of samples from appropriate sources that incorporate adequate features and attributes and implement the data set requirements. "For supervised and unsupervised learning, a data set is used to train, test and verify an algorithm using ML techniques. The part of the data set used to develop the algorithm is referred to as training data; the part used for testing is test data. Both training data and test data are used by the development team. A separate, possibly overlapping, data set, termed verification data may be used for assurance, independent of that team. Note that these definitions apply when the data is used as part of a pre-deployment training and development phase, as well as when there is continual learning." [SCSC-153B]
Epistemic uncertainty	"uncertainty caused by a lack of knowledge of the physical world (knowledge uncertainty)."[81]
Equivalence class	For this paper, an equivalence class can be a segmenting of the operational design domain (ODD), and the data set, or the performance of the ML model for those specific segments, where they are equivalent to other segments of the ODD, data set and ML model.

[80] Fong, Terry, "Autonomous Systems", National Aeronautics and Space Administration 2018, [URL: https://www.nasa.gov/sites/default/files/atoms/files/nac_tie_aug2018_tfong_tagged.pdf]

[81] Pereira A, Thomas C. Challenges of Machine Learning Applied to Safety-Critical Cyber-Physical Systems. Machine Learning and Knowledge Extraction. 2020; 2(4):579-602. [URL: https://doi.org/10.3390/make2040031].

Term	Definition
Generalization	Refers to the Machine Learning inference model's ability to make correct predictions on new, previously unseen data as opposed to the data used to train the Machine Learning model during development[82]
Holdout data	"Examples intentionally not used ("held out") during training. The validation dataset and test dataset are examples of holdout data. Holdout data helps evaluate your model's ability to generalize to data other than the data it was trained on. The loss on the holdout set provides a better estimate of the loss on an unseen dataset than does the loss on the training set."[83]
Hyperparameter (Machine Learning)	"A setting that is used to control the behavior of a learning algorithm (e.g., number of hidden layers, learning rate, number of neurons per layer)."[84]
Independence	For ML data sets, separate samples used to formulate training, validation, and test data sets should be unique from one another, i.e., holdout data sets.
Inference Model	The ML software item to be provided for sub-systems integration, which may be associated with a parameter data item containing hyper parameter values.
Item	An item is a grouping of components that are able to accomplish one or more functions and are a single configuration item. An item can either be hardware or software.
Labelling	"In supervised learning, algorithms learn from a dataset of examples labeled with an output variable representing the right answer. Thus, the data should be collected and preserved in a way that avoids corruption of the labels. Each example in the training dataset should be checked to assure [*sic*] that it has an associated output (label). " [AFE-87]
Leakage	"where the algorithm has access to information that should not legitimately be available" [SCSC-153B]
Locked	The inference model is not allowed to update the weights of the ML model while fielded/deployed. Within this document, locked is synonymous with frozen, non-adaptive, and static, while antonyms are online, continuous, unlocked, and adaptive.
Machine Learning	A subfield of computer science where an algorithm is trained on a data set to produce a model that can be used to predict future values.[85,86]
Machine Learning Architecture	The design considerations used to create the model, which include considerations like type of model and format of data used.
Machine Learning Model	A Machine Learning algorithm that will be trained, validated, and verified on a data set, and implements the Machine Learning model requirements.
Model	(1) a representation of something [Merriam-Webster] (2) "The representation of what a Machine Learning system has learned from the training data." [87]

[82] Google Machine Learning Glossary of Terms, [URL: https://developers.google.com/machine-learning/glossary].
[83] [URL: https://developers.google.com/machine-learning/glossary#machine-learning, Accessed: 10/14/2022]
[84] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, The MIT Press, 2016.
[85] Arora, S., "Mathematics of Machine Learning: An introduction", Princeton University Computer Science, Institute for Advanced Study, Plenary talk at International Congress of Mathematicians 2018, [URL: https://unsupervised.cs.princeton.edu/ICMtalk/aroraplenary.html, Accessed: 10/7/2022]
[86] Mohri, M., Rostamizadeh, A., and Talwalkar, A., Foundations of Machine Learning, Second Edition, MIT Press, Dec 25, 2018. [URL: https://cs.nyu.edu/~mohri/mlbook/, Accessed: 8/23/2022]
[87] Google Machine Learning Glossary of Terms, [URL: https://developers.google.com/machine-learning/glossary].

Term	Definition
Model Based Development (MBD)	As described in RTCA DO-331, MBD uses an appropriate model notation to aid in the design, implementation, and development of the software item. Refer to DO-331 for definitions related to MBD.
Non-Data Driven Collected Data Sets	Data sets not collected based on systematic, data-driven methodologies.
Operational Design Domain (ODD)	For an item, the ODD is the environment in which an item operates and is defined by the system process. The item may know of the ODD through input interfaces.
Overfitting	"Creating a model that matches the training data so closely that the model fails to make correct predictions on new data. Regularization can reduce overfitting. Training on a large and diverse training set can also reduce overfitting."[88]
Parameter data item	A configuration file that can change the performance of a software item.
Representativeness (Data Set)	"Training data should contain all foreseen scenarios in which the system will be used; training data should contain a representative number of rare examples; features in the dataset should be selected or generated from raw features according to their relevance to the function that the ML model will perform." [AFE-87]
Reinforcement learning	An ML learning approach that optimizes a reward function, which indicates the quality of an action taken at each iterative step.
Robustness	The ability of the item to operate in adverse conditions, which may be corner cases or edge cases.
Stability	ML model remains unchanged (or only slightly changed) in the presence of perturbation inputs.
Sufficiency	"Data should be sufficient in quantity (statistical significance), which will depend on the problem domain. In general, the more good-quality data, the better; but processing more data comes with a price of computational burden. In order to assure data quality, a data curation process is good practice, i.e., to perform data organization, integration, and management through its life cycle in order to make it useful for data-driven uses such as ML applications." [AFE-87]
Supervised learning	An ML data-driven learning approach that utilizes a labelled data set.
Subsystem	A further division of the system into smaller logical sub-areas of specialty, e.g., FADEC (Full Authority Digital Engine Control) subsystem for the propulsion system.
System	A portion of the aircraft or UAS normally divided along domains, e.g., propulsion, navigation, flight.
Trace data	The artifact that substantiates the bi-directional linkage between a parent artifact and a child artifact.
Training- [ML process]	Training: "The process of determining the ideal parameters comprising a model."[89]

[88] [URL: https://developers.google.com/machine-learning/glossary#overfitting, Accessed: 10/22/2022]
[89] [URL: https://developers.google.com/machine-learning/glossary#training, Accessed: 10/14/2022]

Term	Definition
Testing	"evaluation of the system under various conditions and observing its behavior while watching for defects"[90]
Uncertainty	"Uncertainty is connected with safety when the outcome of a system is unknown, and its probability distribution is also not known (or only partially Known)[91]"[92]
Underfitting	"Producing a model with poor predictive ability because the model hasn't fully captured the complexity of the training data. Many problems can cause underfitting, including Training on the wrong set of features. Training for too few epochs or at too low a learning rate. Training with too high a regularization rate. Providing too few hidden layers in a deep neural network."[93]
Unstructured and unlabeled data sets	These data sets do not provide the necessary structure for labeling for supervised learning algorithms.
Unsupervised Learning	An ML learning approach that does not typically leverage labeled training data, i.e., typically uses an unlabeled data set.
Weights	In a neural network, a coefficient/parameter that transforms the feature (input) to the output.
Validation [ML training process]	"A process used, as part of training, to evaluate the quality of a machine learning model using the validation set. Because the validation set is disjoint from the training set, validation helps ensure that the model's performance generalizes beyond the training set."[94]
Verification	"producing a compelling argument that the system will not misbehave under a broad range of circumstances"[95] The process of conducting test, inspection, analysis or other appropriated methodology to ensure the requirements are fulfilled.

[90] Pereira A, Thomas C. Challenges of Machine Learning Applied to Safety-Critical Cyber-Physical Systems. Machine Learning and Knowledge Extraction. 2020; 2(4):579-602. [URL: https://doi.org/10.3390/make2040031].
[91] Möller, N.; Hansson, S.O. Principles of engineering safety: Risk and uncertainty reduction. Reliab. Eng. Syst. Saf. 2008, 93, 798–805.
[92] Pereira A, Thomas C. Challenges of Machine Learning Applied to Safety-Critical Cyber-Physical Systems. Machine Learning and Knowledge Extraction. 2020; 2(4):579-602. [URL: https://doi.org/10.3390/make2040031].
[93] [URL: https://developers.google.com/machine-learning/glossary#underfitting, Accessed: 10/20/2022]
[94] [URL: https://developers.google.com/machine-learning/glossary#validation, Accessed: 10/14/2022]
[95] Pereira, A.; Thomas, C. Challenges of Machine Learning Applied to Safety-Critical Cyber-Physical Systems. Mach. Learn. Knowl. Extr. 2020, 2, 579-602. [URL: https://doi.org/10.3390/make2040031].

Annex C Background of the Document

As indicated by SAE Aerospace Information Report (AIR) 6988[96] and AVSI AFE-87[97], the traditional framework of aviation certification guidance standards is not adequate for the uncertainty added by the probabilistic development techniques used by ML. Moreover, AIR 6988 identifies several gaps associated with meeting the traditional aviation software standard guidance provided by RTCA DO-178C. Some of those gaps are in the area of development of low-level requirements, verifiability of the ML model, ML development lifecycle, and structural coverage analysis.

SAE AIR 6988 then lays out an approach for establishing a framework for AI/ML. This document follows the guidance provided by SAE AIR 6988 in establishing a framework, e.g., limit to ML, offline/frozen learning, and addressing the gaps in the current aviation certification guidance.

[96] SAE AIR 6988™, Issued 2021-04, Artificial Intelligence in Aeronautical Systems: Statement of Concerns

[97] Final Report, AFE 87 - Machine Learning, AFE 87 Project Members, 7 May 2020, Version 1.0, 87-REP-01, Public, Issued 5 June 2020 [URL: https://avsi.aero/wp-content/uploads/2020/06/AFE-87-Final-Report.pdf].

Annex D Document Contributors

Name	Organization	Role	Contact Info
H. Glenn Carter	US Army DEVCOM AvMC SRD	Sponsor	usarmy.redstone.devcom-avmc.mbx.amr-sw@army.mil
Alexander Chan	US Army DEVCOM AvMC SRD	Alternate Sponsor	usarmy.redstone.devcom-avmc.mbx.amr-sw@army.mil
Chris Vinegar	US Army DEVCOM AvMC SRD	Alternate Sponsor	usarmy.redstone.devcom-avmc.mbx.amr-sw@army.mil
Victor Terres	US Army DEVCOM AvMC SRD	Alternate Sponsor	usarmy.redstone.devcom-avmc.mbx.amr-sw@army.mil
Jason Rupert	MTSI	Initial Author	jason.rupert@mtsi-va.com
Joel Craig	WaveLink, Inc.	Advisor/Reviewer	
Allen Scales	Torch, Inc.	Advisor/Reviewer	allen.scales@torchtechnologies.com
Shreeder Adibhatla	MTSI	Advisor/Reviewer	shreeder.adibhatla@mtsi-va.com
Ivan Fernandez	MTSI	Advisor/Reviewer	ivan.fernandez@mtsi-va.com
Deirdre Scully	MTSI	Advisor/Reviewer	deirdre.scully@mtsi-va.com
Jay Ball	US Navy NAVORDSAFSECACT IH	Reviewer	
Ben Werner	US Army DEVCOM Armaments Center	Reviewer	usarmy.redstone.devcom-avmc.mbx.amr-sw@army.mil
Adam Thomas	US Army DEVCOM AvMC SRD	Reviewer	usarmy.redstone.devcom-avmc.mbx.amr-sw@army.mil
Jeff Obermark	US Army DEVCOM AvMC SRD	Reviewer	usarmy.redstone.devcom-avmc.mbx.amr-sw@army.mil
Rachel O'Kraski	US Army DEVCOM AvMC SRD	Reviewer	usarmy.redstone.devcom-avmc.mbx.amr-sw@army.mil
Phi Pham	US Army DEVCOM AvMC SRD	Reviewer	usarmy.redstone.devcom-avmc.mbx.amr-sw@army.mil

Annex E Frequently Asked Questions and Discussion Papers

Annex E-1 Considering Reinforcement Learning

Several programs indicated they desire to field ML data-driven systems where reinforcement learning was used. The assurance approach in this document only considered the use of supervised learning. Future documents will include assurance approaches for the use of reinforcement learning. Some guidance from the assurance community is coming out to specifically address reinforcement learning. Still more considerations related to the selection of reward functions and scenario-based training are necessary before adequate guidance can be provided. There is a hope that much of the guidance provided above for supervised learning will be applicable, but a full evaluation is necessary to determine if that hope is valid.

Annex E-2 Using Formal Methods

Dr. Laura Humphrey[98], from Air Force Research Lab, indicated that the use of formal methods may be possible in the ML lifecycle. Research is still being conducted on proper and useful applications of formal methods to ML lifecycle, specifically the verification process. A survey of applications was conducted by Dr. Humphrey and should be leveraged when considering the possible use of formal methods. In most cases, the use of formal methods involves the use of tools associated with a formal grammar/notation. When considering formal methods, the grammar/notation and tools need to be evaluated on a case-by-case basis to determine if the grammar and tool maturity is appropriate for the application. Follow-on work will continue to examine where formal methods may be used to address the objectives and activities listed above.

Annex E-3 Concerns with adaptive ML

Everything associated with adaptive learning should make the assurance community nervous, and thankfully the adaptive learning mishaps published have been of minimal severity[99,100]. At this time adaptive learning, i.e., continuous learning and updating of the hyperparameters while fielded, is discouraged. This ML data-driven approach is discouraged because of the potential safety impacts through the emergence of unexpected and undesired behaviors.

The assurance community is beginning to explore the necessary processes, objectives, activities, and output data items to ensure adaptive ML can be applied in flight and safety-critical applications. However, currently the recommendations are inadequate to provide assured safe deployment of adaptive learning.

---

[98] Humphrey, Dr. Laura, Team Lead, V&V of Complex & Autonomous Systems, "Formal Methods for V&V and Certification", Aerospace Systems Directorate, Autonomous Controls Branch (AFRL/RQQA), October 2019, https://cra.org/ccc/wp-content/uploads/sites/2/2019/10/Laura-Humphrey.pdf.

[99] https://russell-davidson.arts.mcgill.ca/e706/gaming.examples.in.AI.html, e.g., "Reinforcement learning agent goes in a circle hitting the same targets instead of finishing the race".

[100] "In normal RL, you would pick the collision fine at the beginning of training and keep it fixed forever. The problem here is that if the pay-per-trip is high enough, the agent may not care whether it gets in lots of collisions (as long as it can still complete its trips). In fact, it may even be advantageous to drive recklessly and risk those collisions in order to get the pay. We have seen this before when training unconstrained RL agents." https://openai.com/blog/safety-gym/.

For example, SCSC-153 provides the two architecture level objectives, i.e., ML model design considerations, in Table 3 that require the applicant account for adaptations occurring in the architecture and complex software computation/algorithm. These two objectives are not adequate to endorse adaptive learning in flight and safety-critical applications; however, they are highly valuable because they provide initial indications of the types of objectives necessary to ensure the safe and airworthy deployment of adaptive algorithms.

Table 3. Adaptive Objectives.

ARC3-1: Inappropriate or unauthorised adaptations do not occur. ARC3-2: Computation behaviour is appropriate before, during and after an adaptation.

Including objectives to account for adaptation is one of the significant additions of SCSC-153A over EASA Level 1, which assumes the ML is locked/frozen. The potential inclusion and allowance of adaptation is a game changer. It allows the complex software's function, performance, and other critical attributes to change over time without re-entering the ML-based software item lifecycle. Adaptation allows the complex software to assess its performance and determine that parameter updates are necessary. After an update is performed, the complex software determines that the update improves performance with respect to an onboard desired goal. Techniques, like unsupervised[101] reinforcement learning[102], could allow the computation/algorithm to continuously assess its performance against goals, and understand what modifications within the computation/algorithm or framework are necessary to refine performance to reach a goal. Adaptive learning may better account for real (fielded) data distribution shift and boundary performance concerns; however, the algorithm itself will be determining whether to adapt and could be incorrectly adapting to adverse and adversarial data. Moreover, complex software is susceptible to myopically focusing on singular tasking and characteristics and forgetting critical lessons previously learned during adaptation. Given this, there might be concerns that the adaptation is forgetting previously learned behaviors in favor of more optional performance and not remembering previously learned behaviors[103,104]. All these concerns, and others, e.g., the updating process, must be accounted for when adaptive learning is allowed.

Beginning to consider the possibility of allowing adaptive learning starts with the algorithm justifying the update is appropriate and authorized. Immediately undermining the assurance case would occur if the computation or framework attempted an unauthorized and inappropriate update. An example of an inappropriate update is one where the update would dramatically reduce the safety margin of the

[101] "Unsupervised Learning is an application of AI, which does not typically leverage labeled training data. Instead, algorithms are tasked with identifying patterns in data sets on their own by defining signals and potential abnormalities." ANSI/CTA Standard Definitions and Characteristics of Artificial Intelligence, ANSI/CTA-2089.
[102] "Reinforcement Learning is defined as an application of AI, in which information is learned through a training method based on rewarding desired behaviors or punishing undesired ones." ANSI/CTA Standard Definitions and Characteristics of Artificial Intelligence, ANSI/CTA-2089.
[103] Overcoming Catastrophic Forgetting for Continual Learning Via Model Adaptation, Wenpeng Hu, et al., ICLR 2019, https://openreview.net/pdf?id=ryGvcoA5YX. [45 citations]
[104] Overcoming catastrophic forgetting in neural networks, James Kirkpatrick, et al., Proceedings of the National Academy of Sciences Mar 2017, 114 (13) 3521-3526; DOI: 10.1073/pnas.1611835114, https://www.pnas.org/content/114/13/3521. [2053 citations]

system, and an example unauthorized update could occur during flight or an unauthorized mode of flight.

The computation and framework are expected to demonstrate appropriate and authorized updates that could occur during training, testing, and validation. During that time, the computation and framework would demonstrate an ability to follow its evaluation criteria prior to allowing updates. The computation and framework would be expected to demonstrate examples of approved and rejected adaptions to validate the criteria [105,106]. Especially important is establishing the criteria for rejection and the evaluation process the algorithm uses. Once an adaptation is approved, the computation and framework will go through a process to load the new adaptation. The adaptation process should be thoroughly documented in requirements and design so all users (human or other) will be aware of platform's status during the adaption update. Once the adaption is complete, the platform should evaluate the update to determine that the update was successful. The platform should also check that the adaption is indeed beneficial and does not reduce safety margin of the flight and safety-critical platform. The process for evaluating the update should not place the platform or the surrounding environment in harm or jeopardy when evaluating the updated adaption. Should the adaption be found to be harmful to the safety of the platform and surroundings the platform should be able to place itself in a safe state and roll back to previously safe adaption and cause no harm. SCSC-153A calls additional attention to the training necessary for personnel interacting with an adaptive system. They will need to be made aware of the changes to the system performance since the last interaction and how to interact with the updated system safely.

Due to the possibility of the adaptation failing SCSC-153A stresses: "every platform ought to be capable of being put in a safe state that can be maintained for a considerable period." To accomplish, this SCSC-153A recommends "instantiate two (or more) autonomy architectures" to allow a voter or other mechanism to enable the assured safe version to control while the other adaptions remain on standby until proven to be appropriately safe.

Going forward, we will continue to explore the concepts associated with adaptive learning, as these two objectives alone do not cover the extensive complexity and possible computations associated with being able to ensure adaption is applied safely to flight and safety-critical applications. Further exploration will allow the concerns to be appropriately captured, and objectives established to provide the appropriate assurances from all levels (platform, architecture, and computation) that adaption will improve the system's performance while maintaining airworthiness concerns. Moreover, the complex software field should produce representative approaches demonstrating the artifacts, processes, and constraints[107] necessary to ensure unbounded unsafe adaptations cannot occur.

[105]A comprehensive survey on safe reinforcement learning, J. Garcia and F. Fernández. Journal of Machine Learning Research, 16 (2015), pp. 1437–1480. https://www.jmlr.org/papers/volume16/garcia15a/garcia15a.pdf. [745 citations]

[106] Assured Reinforcement Learning with Formally Verified Abstract Policies, G. Mason, et al., 2017. https://pure.york.ac.uk/portal/files/50943955/ICAART_2017_GRM.pdf.

[107] Examples of constrained reinforcement algorithms: Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), Lagrangian penalized versions of PPO and TRPO, and Constrained Policy Optimization (CPO).

Annex E-4 Concerns with Interpreted Programming Languages

Interpreted programming languages produce byte code which requires a virtual machine interpreter to be able to execute the byte code on the target hardware. Examples of interpreted languages are Java and Python. DO-178C discusses machine code, compilers, and executable object code and defines executable object code as the following: "A form of code that is directly usable by the processing unit of the target computer and is, therefore, a compiled, assembled, and linked binary image that is loaded into the target computing hardware." While DO-178C does not explicitly exclude interpreted programming languages, DO-178C established objectives and activities for the environment (methods, tools, procedures, programming languages, and hardware) such that they "limit the opportunity for introducing errors." No precedent exists for the deployment of interpreted languages on flight-critical applications, i.e., applications that have a high development assurance level (DAL), nor is the possibility of appropriately certifying/qualifying an interpreted programming language, especially Python, something likely to occur in the near term, e.g., the next 5-10 years. Instead, if a program is using an interpreted programming language during the MLDL, they should plan to transition to a compiled program language in the MLIL. For more information on this topic reference: "Concerns with using Python in Machine Learning Flight Critical Applications"[108].

[108] H. Glenn Carter, Alexander Chan, Chris Vinegar, Jason Rupert, "Concerns with using Python in Machine Learning Flight Critical Applications". Presented at Vertical Flight Society (VFS) Forum 79 Conference, 2023, DOI 10.4050/F-0079-2023-18015.