White Paper
June 2026

AI Requirements Framework

for Safety-Critical Systems

Requirements and Verification Standards for Artificial Intelligence in Safety-Critical Applications

Kevin Williams

SAFETY CRITICAL LABS

Abstract

Artificial intelligence is being deployed in safety-critical systems across aerospace, automotive, medical, and industrial domains at a pace that outstrips the standards governing them. The foundational software engineering standards that protect human life and mission success (DO-178C for aviation, ISO 26262 for automotive, NPR 7150.2D for NASA systems) were designed for deterministic, logic-based software. They assume that faults are bugs in code, that verification happens once at delivery, that outputs are deterministic, and that decision logic can be traced through code inspection. AI and ML systems violate every one of these assumptions.

This white paper presents the AI Requirements Framework for Safety-Critical Systems, a domain-agnostic standard, now at Version 3.0, that supplements existing standards with AI-specific requirements and co-located verification approaches. The framework is organized into two tiers: ten core requirement areas (AI-1 through AI-10) that apply to every AI system regardless of underlying technology, and three architecture- and paradigm-specific requirement areas (AI-11 through AI-13) that apply when a system employs multi-model coordination, neural network architectures, or continuous learning. Throughout, the framework follows a foundational principle: requirements should be defined by the failures they prevent, not the capabilities they enable. Each requirement area addresses a specific, documented AI failure mode and includes verification methods and success criteria traceable to that failure mode.

The paper demonstrates that AI systems in safety-critical applications require purpose-built requirements that assume non-deterministic behavior, continuous performance variability, and the possibility of confidently wrong outputs, and that these requirements must be verifiable, auditable, and integrated into existing software engineering lifecycles. The framework is published with a persistent digital object identifier (DOI: 10.5281/zenodo.19501092) and was developed in the context of certification assessment of AI components for government human spaceflight programs.

Table of Contents

PART I

The Problem

1. Introduction

Traditional software behavior is explicitly defined by code. An engineer writes instructions; the system follows them. Verification confirms the code implements the required logic, and once verified, behavior remains stable unless the code is deliberately changed. This deterministic relationship between code and behavior is the foundation upon which every safety-critical software standard in use today has been built.

AI and ML systems operate on a fundamentally different principle. Their behavior emerges from patterns learned during training rather than from explicitly programmed instructions. A neural network that classifies images does not contain a line of code that says "if the image contains a tumor, return positive." Instead, it has learned statistical patterns across thousands of labeled examples, encoding that knowledge in millions of numerical weights that no human can meaningfully inspect. The behavior is real, often remarkably accurate, but it is emergent, not prescribed.

This distinction has profound implications for safety-critical systems. Code inspection alone cannot verify AI behavior, because there is no code that encodes the decision logic. Validated performance may not persist over time, because the relationship between inputs and correct outputs can shift. And outputs may be non-deterministic: identical inputs processed at different times may produce subtly different results.

These characteristics do not make AI systems unsuitable for safety-critical operations. They do, however, require verification approaches designed for data-driven, emergent behavior, approaches that the standards governing safety-critical software today were never designed to provide.

This framework supplements, does not replace, existing software and model assurance requirements. It addresses AI-specific failure modes not covered by traditional software or model and simulation verification practices, reconciles AI model validation with established credibility assessment approaches, and provides the necessary rigor for AI systems impacting human safety and operational success.

2. The Assumption Gap

Every safety-critical software standard in use today was developed for a specific type of system: deterministic, logic-based software whose behavior is explicitly defined by code. For the systems they were designed to govern, they work extraordinarily well. DO-178C has contributed to making commercial aviation the safest mode of transportation in human history. ISO 26262 has driven measurable improvements in automotive functional safety. NPR 7150.2D provides rigorous software assurance for NASA's most critical missions.

The problem is not that these standards are wrong. The problem is that they are built on assumptions that AI/ML systems violate.

2.1 Structural Gap Analysis

Through analysis of the foundational assumptions embedded in current standards, six structural gaps emerge when these standards are applied to AI/ML systems.

Traditional Assumption	AI/ML Reality	Risk If Unaddressed
Fault = bug in code	Fault = bias, drift, hallucination, data quality issues	Wrong failure mode analysis
V&V performed at delivery	Requires continuous validation throughout operations	Post-deployment failures undetected
Requirements trace to code	Behavior emerges from training data	Loss of traceability
Deterministic outputs	Probabilistic outputs with confidence distributions	Unpredictable edge-case behavior
Code coverage metrics apply	No equivalent for trained models	Inadequate test coverage; false assurance
Decision logic is explicit and inspectable	Decision logic encoded in learned weights	Operators cannot verify AI reasoning

Table 1. Structural gaps between traditional software standards and AI/ML system characteristics.

2.2 The Compounding Effect

These gaps do not exist in isolation. They compound. Consider a system with probabilistic outputs that encounters out-of-distribution inputs while operating under a standard that assumes static post-deployment behavior and tests only with code coverage metrics. That system has no mechanism to detect failure across any of these dimensions. The standard certifies the system as safe. The system deploys. The system fails, confidently, silently, and in ways the certification process was structurally incapable of anticipating.

The compounding effect is what makes incremental patches to existing standards insufficient. Each gap reinforces the others: a system that cannot detect drift (Gap 2) will not recognize when its probabilistic outputs (Gap 4) are becoming unreliable on inputs it was never trained for (Gap 6), and because there is no code to inspect (Gap 1), the failure will not be caught through traditional verification (Gap 5). The operator, lacking any explanation of the AI's reasoning (Gap 6), may trust a confidently wrong output until consequences make the failure undeniable.

2.3 Why Patching Existing Standards Is Insufficient

The natural institutional response to these gaps has been to add AI-specific clauses to existing standards or to publish guidance documents suggesting best practices. These efforts, while well-intentioned, are structurally insufficient for three reasons.

The verification architecture is wrong. Existing standards are built around a verify-at-delivery paradigm. AI systems require continuous verification. Adding a clause that says "consider monitoring model performance" to a standard whose entire verification framework assumes one-time certification does not create a functioning continuous validation system. It creates a suggestion without enforcement, accountability, or defined evidence artifacts.

The failure model is wrong. Existing standards define failure in terms of code defects. AI failures (bias, hallucination, drift, out-of-distribution confidence) are not code defects. They are emergent properties of data-driven systems. A bias in hiring recommendations is not a bug that can be found through code inspection. A hallucinated medical diagnosis is not a null pointer exception. These failure modes require fundamentally different detection mechanisms, and standards that do not recognize them as distinct categories of failure will not test for them.

The trust model is wrong. Existing standards assume that verified software can be trusted within its operational envelope. AI systems can produce confidently wrong outputs within their nominal operating envelope. A classification model reporting 97% confidence may be encountering an input far outside its training distribution, a scenario in which its confidence score is meaningless. The concept of operator trust calibration, ensuring humans know when to trust AI and when to apply scrutiny, has no equivalent in any current safety-critical software standard.

2.4 Gap-to-Requirement Mapping

Each gap identified above maps directly to a specific requirement area in the framework. This traceability ensures that no identified gap remains unaddressed and that every requirement exists to close a documented gap.

Gap / Risk	Framework Requirement Area
Training data integrity and traceability	AI-1: Operational and Data Foundations
Bias and fairness issues	AI-2: Addressing AI Bias
No test coverage equivalent for trained models	AI-3: ML Test Coverage
Static after deployment; silent degradation	AI-4: Continuous Validation
Unpredictable, fabricated outputs	AI-5: Hallucination Prevention
Vulnerable to novel out-of-distribution inputs	AI-6: Out-of-Distribution Detection
Vulnerable to adversarial attacks	AI-7: Adversarial Robustness
Black-box decision making	AI-8: Explainability
Operators cannot verify reasoning	AI-9: Human-AI Teaming
Personal data recoverable from trained models; automated decisions evade data subject rights	AI-10: Privacy and Data Protection

Multi-model coordination failures: error propagation, correlated failures, combined-confidence misrepresentation	AI-11: Multi-Model Systems
Neural-network-specific failure modes: confidence miscalibration, opacity, adversarial brittleness, generative hallucination	AI-12: Neural Networks
Continuous learning risks: catastrophic forgetting, learning instability, unbounded ODD evolution	AI-13: Continuous Learning and Adaptation

Table 2. Gap-to-requirement mapping.

The first ten requirement areas form the framework's core and apply to every AI system the framework certifies. The final three apply conditionally, based on the system's architecture and learning paradigm, following the two-tier structure described in Section 3.3.

PART II

The Framework

3. Framework Design Principles

3.1 Foundational Principle: Failure-Defined Requirements

Every requirement in the framework is defined by the failure it prevents, not the capability it enables. This principle has three practical consequences.

Completeness through failure coverage. The set of requirements is derived from the set of known AI failure modes. If a failure mode exists and is not covered, it represents a gap in the framework. This makes completeness assessable: rather than asking "have we thought of everything?" the question becomes "is there a known AI failure mode not addressed by at least one requirement?"

Verifiability through failure testing. Each requirement's verification includes testing against known failure cases: injected hallucinations, adversarial inputs, out-of-distribution scenarios, and boundary conditions. If a requirement cannot be tested by simulating the failure it prevents, the requirement is not sufficiently defined.

Relevance through operational grounding. Requirements are defined in terms of operational impact, not model architecture. The framework does not prescribe whether a project uses neural networks, decision trees, or ensemble methods. It prescribes what protections must be in place regardless of architecture.

3.2 Applicability Criteria

Before requirements apply, the framework requires a determination of AI/ML presence. A system is considered to incorporate AI/ML if it meets one or more of the following criteria: its behavior is derived from training data rather than explicitly programmed logic; its outputs are probabilistic or non-deterministic; its behavior may change or degrade over time without code modifications; or it makes or informs decisions affecting human safety or operational success.

These criteria are deliberately broad. A lookup table is not AI. A Kalman filter with fixed parameters is not AI. But a system that uses a neural network to classify sensor data, a reinforcement learning agent that adapts its strategy, or a natural language model that generates recommendations; each of these meets one or more criteria and falls within the framework's scope.

A system meeting one or more criteria proceeds to classification. A system meeting no criteria falls under standard software assurance practices, and this framework does not apply.

3.3 AI System Classification and the Two-Tier Requirement Structure

Version 3.0 organizes the framework's requirements into two tiers. **Core Requirements** (AI-1 through AI-10) apply to every AI system the framework certifies, regardless of underlying technology. **Architecture and Paradigm Requirements** (AI-11 through AI-13) apply conditionally, based on the system's design choices: AI-11 where two or more AI models interact, AI-12 where neural network architectures are employed, and AI-13 where the system learns or adapts during operation. Architecture refers to structural design choices (multi-model coordination, neural network use, ensemble methods) that introduce failure modes beyond those addressed by the algorithm-agnostic core. Paradigm refers to learning and operational patterns (continuous learning, online adaptation) that change how the system behaves over time.

Once identified as meeting applicability criteria, systems are classified into one of three tiers. Classification determines which requirements apply and the level of tailoring authority available.

Classification	Description	Applicable Requirements
Safety-Critical AI	AI outputs directly affect human safety or system survivability	All ten core requirement areas (AI-1 through AI-10), plus AI-11, AI-12, and AI-13 wherever the corresponding architecture or paradigm is employed; no tailoring without PSRB approval
Mission-Critical AI	AI outputs affect operational success but not human safety	AI-1 through AI-6, AI-8, AI-9, and AI-10, with AI-7 tailored; architecture and paradigm requirements as applicable; tailoring permitted with Chief Engineer approval
Operational Support AI	AI supports operations but does not drive critical decisions	AI-1 through AI-6 and AI-10 minimum, with AI-7, AI-8, and AI-9 as tailored; architecture and paradigm requirements as applicable; tailoring permitted with SA authority approval

Table 3. AI system classification and requirement applicability.

The classification tiers are not arbitrary. Safety-Critical AI requires the full complement of requirements because a failure in any dimension (bias, drift, hallucination, adversarial compromise, privacy breach, or operator miscalibration) could result in harm. Mission-Critical AI permits tailoring because operational failures, while consequential, do not threaten lives. Operational Support AI requires baseline protections but allows the most flexibility in implementation.

Two additional rules bound the classification decision. A system that can change or degrade over time without code modification, meeting the potential-drift criterion, is classified no lower than Mission-Critical, and any tailoring of its requirements requires documented approval. And AI-10 applies wherever the system processes personal data under any applicable privacy framework; where it does not, AI-10 sub-requirements are documented as Not Applicable with rationale, recorded in the system's operational design domain declaration.

3.4 Integration with Existing Lifecycles

The framework does not create a parallel lifecycle. It identifies AI-specific activities that must be performed within each phase of the existing software lifecycle: requirements, design, implementation, test, and verification. During requirements, projects define AI function criticality and identify protected classes and operational contexts. During design, they establish model architecture and explainability approaches. During implementation, they partition datasets and document provenance. During test and verification, they execute AI-specific test matrices and compile evidence packages.

This integration approach means organizations do not need to abandon their existing processes. They need to augment them with activities that address the AI-specific failure modes their current processes were not designed to catch.

3.5 Tailoring Guidance

Tailoring authority scales with classification. Safety-Critical AI permits no tailoring without Project Safety Review Board approval, a deliberate constraint reflecting the consequence of getting it wrong. Mission-Critical AI permits tailoring with Chief Engineer approval and documented rationale. Operational Support AI permits tailoring with Software Assurance authority approval. In all cases, tailoring decisions must be documented with rationale and retained as verification evidence. The principle is simple: the higher the stakes, the higher the authority required to deviate from the framework.

3.6 Prescriptive Boundaries, Flexible Implementation

A critical design choice governs the framework's structure: it is prescriptive about failure modes and flexible about implementation methods. This is not a compromise; it is a deliberate regulatory architecture.

The framework prescribes what must be protected against. Every AI system classified as Safety-Critical must detect out-of-distribution inputs, must prevent hallucinated outputs from driving autonomous action, must provide operators with calibrated confidence indications, and must maintain human decision authority over safety-critical actions. These are non-negotiable requirements derived from documented failure modes. They do not vary by algorithm, architecture, or implementation approach.

The framework does not prescribe how those protections are implemented. A project using a convolutional neural network for image classification and a project using a gradient-boosted ensemble for sensor fusion face different technical realities. The framework requires both to detect out-of-distribution inputs; it does not mandate that both use Mahalanobis distance or ensemble disagreement or any particular detection method. It requires both to provide explainability; it does not mandate that both use SHAP, LIME, or attention visualization.

This distinction operates at three levels:

Level	Prescriptive (Fixed)	Flexible (Project-Defined)
Failure Modes	Which failure modes must be addressed (all ten core requirement areas for Safety-Critical AI, plus applicable architecture and paradigm requirements)	None
Requirements	What protections must be in place (e.g., OOD detection must exist, confidence must be calibrated)	How protections are implemented (detection algorithms, calibration methods, explanation techniques)
Thresholds	Which threshold categories must be documented (performance, confidence, drift, OOD, escalation)	What threshold values are appropriate (derived from project risk tolerance and operational context)

This architecture ensures the framework remains relevant as AI technology evolves. New model architectures, training paradigms, and deployment patterns will emerge. The failure modes they face (bias, drift, hallucination, adversarial vulnerability, opacity) will persist. By anchoring requirements to failure modes rather than implementation details, the framework avoids the obsolescence that plagues technology-specific regulation while maintaining the enforceability that principles-based guidance lacks.

The practical consequence for projects is clear: you must demonstrate that each applicable failure mode is addressed with verifiable protections and auditable evidence. You choose the technical approach. The framework evaluates the outcome.

PART III

Core Requirements: Addressing AI Failure Modes

4. Operational and Data Foundations (AI-1)

THE FAILURE MODE

Traditional software has direct traceability from requirements to code. An engineer can inspect the code and verify it implements the requirement. AI systems break this traceability because behavior emerges from training data rather than explicit logic. If the training data is flawed (contaminated, biased, mislabeled, or improperly partitioned), the resulting model will encode those flaws into its behavior, and no amount of code inspection will reveal the problem.

Consider a medical imaging classifier trained to detect tumors. If images from the test set inadvertently leak into training data, a phenomenon called data leakage, the model's performance metrics will be artificially inflated. It will appear to perform brilliantly in evaluation, having effectively memorized the right answers. When deployed on genuinely novel patient scans, its performance may degrade catastrophically, with no external indicator that the evaluation was compromised.

WHAT THE FRAMEWORK REQUIRES

The operational design domain comes first. Version 3.0 anchors this requirement area in a declared operational design domain (AI-1.0): a documented statement of where, when, and for whom the system is validated to operate, including the subject population the system acts upon, the environmental and operational bounds it was validated against, and the regulatory authorizations under which it operates. The ODD declaration is the foundation that later requirements reference: privacy applicability (AI-10) is determined there, and continuous-learning systems must declare whether the ODD itself is permitted to evolve (AI-13). From that foundation, the data requirements follow.

The framework addresses data integrity through four interlocking protections.

Dataset separation requires strict partitioning between training, validation, and test datasets, enforced through technical controls, not just policy. This means separate storage locations, access controls, and automated pipeline checks that prevent data from crossing partition boundaries. Partition integrity must be verified prior to every training run and evaluation cycle.

Partition use restriction goes beyond separation to ensure each partition is used only for its intended purpose. Even with proper separation, using validation data for final testing or repeatedly evaluating against the test set during development compromises the integrity of performance estimates. Access logging and violation alerts provide accountability.

Data provenance requires documentation of the full lineage of every dataset: where the data came from, how it was collected, what transformations were applied, and what preprocessing steps were performed. In safety-critical applications, the inability to trace a model failure back to its data origins makes root cause analysis impossible and retraining decisions uninformed.

Ground truth labeling quality addresses the foundational risk of supervised learning: models learn to predict labels, and if labels are wrong, the model learns wrong behavior with high confidence. For safety-critical applications, the framework requires inter-rater reliability validation (Cohen's Kappa of 0.8 or higher), independent spot-checking of label accuracy, and documented resolution of ambiguous or disputed labels.

VERIFICATION APPROACH

Verification relies primarily on inspection and analysis. Inspectors confirm that partition documentation exists with unique identifiers, that technical controls enforce boundaries, and that provenance records trace to source

data. Analysis confirms that no leakage exists between partitions and that labeling quality metrics meet defined thresholds. The evidence artifacts (partition records, provenance documentation, and labeling quality reports) become part of the permanent verification record.

5. Addressing AI Bias (AI-2)

THE FAILURE MODE

AI systems trained on historical data can encode and amplify biases present in that data, leading to systematic performance disparities across operational conditions or user populations. This is not a theoretical concern; it is a documented pattern across domains. Hiring algorithms have systematically disadvantaged qualified candidates. Facial recognition systems have exhibited dramatically different accuracy rates across demographic groups. Medical risk prediction models have underestimated risk for underserved populations.

In safety-critical applications, the consequences are more severe and more subtle. A sensor fusion system trained predominantly on daytime data may perform poorly in low-light conditions, a bias not against people, but against operational contexts underrepresented in training data. A diagnostic system trained on data from a single hospital may encode that institution's diagnostic patterns, failing when deployed in environments with different equipment, protocols, or patient demographics.

Bias in this framework refers to any systematic deviation in AI performance, whether across environmental conditions, sensor degradation states, or user characteristics. Traditional software verification assumes uniform behavior. AI systems can behave very differently depending on what part of the input space they are operating in, and those differences may correlate with characteristics that matter for safety and fairness.

WHAT THE FRAMEWORK REQUIRES

The bias requirement area is the most expansive in the framework, reflecting the complexity and multi-dimensional nature of the problem. It spans thirteen sub-areas organized around a lifecycle of bias management: prevention, detection, response, and accountability.

Prevention begins before any model is trained. The framework requires bias-free baseline performance established across user classes and operational contexts, providing a reference point against which operational performance can be measured. Training datasets must account for underrepresented groups, historical biases, and labeling errors, and must be validated prior to model development to confirm these requirements are met. Edge-case datasets and stress scenarios must be defined to test bias at operational boundaries where it is most likely to emerge.

Detection requires explicit definition of bias indicators and thresholds for each AI function, continuous monitoring during operations, and impact assessments for all AI decisions affecting human safety, resource allocation, mission planning, and operational procedures. Without defined metrics and thresholds, bias detection becomes subjective and unverifiable.

Response ensures that detection leads to action through a defined enforcement chain that scales with decision criticality.

For non-safety-critical functions, the system must provide bias-corrected decision pathways and alternative recommendations when bias indicators exceed predefined thresholds. Detection without corrective action provides no protection, the system must have predefined responses, not just alerts.

For safety-critical functions, the enforcement chain is absolute and non-negotiable. The framework establishes a four-stage enforcement mechanism:

Prohibition: Bias must not adversely impact safety-critical operations or human safety. This is not an aspiration; it is a constraint. The system must identify every AI decision point that affects safety-critical operations and implement protective action at each one.

Blocking: When bias is detected at a safety-critical decision point, autonomous execution is blocked. The system does not proceed with a biased output and attach a warning. It stops. The biased output does not propagate to downstream systems, does not influence actuators, and does not inform autonomous action.

Mandatory Human Review: The system escalates to a mandatory human review state. The operator receives the detected bias condition, the affected decision context, and the available alternatives. The system cannot resume autonomous execution until the review is complete. There is no timeout that defaults to autonomous continuation.

Documented Override: If the operator determines that the AI output should be used despite the bias detection, the override must be documented with rationale. This creates an auditable record that supports post-mission analysis and accountability. The system override capability must be functional and tested; an override path that exists in design documentation but fails under operational conditions provides no protection.

This enforcement chain is verified through simulation. Test scenarios inject bias conditions at safety-critical decision points and confirm that each stage functions: autonomous execution blocks, mandatory review triggers, the system cannot proceed without human input, and override logs capture decisions and rationale. These are not demonstrations of policy awareness; they are functional tests of safety-critical control paths.

Accountability requires comprehensive logging of bias detection events, corrective actions, and decision rationale. Without logging, bias events cannot be reconstructed for root cause analysis or used to improve subsequent model versions.

VERIFICATION APPROACH

Verification of bias requirements uses the full range of methods. Inspection confirms documentation of baselines, indicators, thresholds, and edge cases. Analysis confirms representation across applicable classes and the absence of known bias patterns, or documents risk assessments with mitigation strategies where absence cannot be confirmed. Testing validates that monitoring systems are operational, alerts are generated when thresholds are exceeded, corrective pathways are invoked, and safety-critical bias escalation functions correctly under simulated bias scenarios.

6. Test Coverage for Data-Driven Systems (AI-3)

THE FAILURE MODE

Traditional software testing relies on code coverage metrics (statement coverage, branch coverage, modified condition/decision coverage (MC/DC)) to demonstrate that the software has been adequately exercised. These metrics are meaningful because traditional software has code that can be covered: every branch, every condition, every statement represents a decision point that testing can exercise.

These metrics are meaningless for ML models. There is no "code" to cover in a neural network's weights. A model with millions of parameters does not have branches to test or conditions to evaluate in the traditional sense. Applying code coverage metrics to an ML system produces a number that satisfies a checkbox without providing any assurance that the model has been adequately tested.

The real danger is not the absence of coverage metrics; it is the false assurance that comes from applying the wrong ones. A project that reports "100% code coverage" for its ML pipeline has tested the data loading, preprocessing, and inference code. It has tested none of the learned behavior that actually determines what the model outputs.

WHAT THE FRAMEWORK REQUIRES

The framework requires projects to define and achieve test coverage metrics appropriate to data-driven systems. These metrics must be justified based on model type and mission criticality, because different models require different coverage approaches. For an image classifier, coverage might focus on variations in lighting, angle, occlusion, and weather conditions. For a time-series predictor, coverage might address temporal patterns, seasonality, anomalies, and phase transitions.

Beyond metric definition, the framework requires testing across representative operational input distributions, the full range of inputs expected during actual operations, including nominal conditions, off-nominal conditions, and transitions between operational phases. Gaps between test and operational distributions represent untested regions where model behavior is uncertain, and must be mitigated through additional data collection, synthetic data generation, or documented risk acceptance.

Boundary and edge case testing addresses the observation that ML models often fail at input boundaries, values at the edges of the training distribution or combinations of inputs not well-represented in training data. These are precisely the conditions most likely to be encountered in safety-critical operations and least likely to be captured by random sampling.

VERIFICATION APPROACH

Verification confirms that coverage metrics are defined and justified, that thresholds are achieved prior to deployment, that operational distributions are characterized and tested, and that boundary and edge case testing is systematic rather than ad hoc. Coverage gaps must be identified and dispositioned, either through additional testing or through documented risk acceptance. The evidence artifacts include coverage metric definitions, threshold justifications, test execution reports, and gap disposition records.

7. Continuous Validation (AI-4)

THE FAILURE MODE

Traditional software standards assume that verification is a one-time event. A system is tested, verified, and deployed. Unless the code changes, the behavior remains stable. This assumption is so deeply embedded in safety-critical standards that most do not even contemplate the possibility of post-deployment performance degradation without a code change.

AI systems violate this assumption routinely. Model performance can degrade during operations due to data drift (the statistical characteristics of production data diverge from training data), concept drift (the relationship between inputs and correct outputs changes), or environmental changes that push the system into operating regimes not well-represented in training data. A sensor degradation that subtly shifts input distributions, a seasonal change in environmental conditions, or a gradual shift in user behavior can all cause a deployed model to become progressively less reliable, with no code change, no warning, and no detection mechanism under traditional standards.

WHAT THE FRAMEWORK REQUIRES

The framework establishes continuous validation as a mandatory operational practice, not a suggested best practice. Five interlocking requirements create a continuous validation system.

Data drift monitoring requires the system to detect when production data diverges from training data distributions using defined statistical methods such as KL divergence or population stability index. Drift metrics must be logged with timestamps, and alerts must be generated when drift exceeds defined thresholds. The question is not whether drift will occur; it is when, and whether the system will detect it before it matters.

Performance threshold maintenance requires continuous computation of performance metrics during operation, with defined mitigation responses when performance degrades below threshold. Performance thresholds must be derived from mission risk tolerance and must be appropriate to the AI function.

Model maintenance criteria ensure that when degradation is detected, predefined response procedures exist. Without documented criteria for when and how to update models, operators may allow degraded performance to continue or apply ad hoc fixes that introduce new risks.

Output validity boundaries provide a safety envelope independent of model internals. AI systems can produce outputs that are technically valid within data type constraints but operationally invalid: physically impossible values, predictions outside safe operating ranges, or results inconsistent with known physical constraints. Boundary enforcement catches erroneous outputs regardless of cause.

Periodic model validation provides a systemic check that the model continues to perform as originally validated. Continuous monitoring of individual metrics may not capture gradual, systemic degradation. Periodic comprehensive validation against baseline scores addresses this risk.

VERIFICATION APPROACH

Verification is primarily test-based. Tests confirm that drift detection mechanisms are operational, performance metrics are continuously computed, degradation triggers defined responses, output boundaries are enforced, and periodic validation executes automatically at specified intervals. The evidence artifacts include drift detection logs, performance trend data, boundary violation records, and periodic validation reports.

8. Hallucination Prevention (AI-5)

THE FAILURE MODE

AI systems, particularly neural networks and generative models, can produce outputs that appear valid but are completely fabricated or contradicted by physical reality. NIST AI 600-1 defines hallucination as "the production of confidently stated but erroneous or false content." The critical word is "confidently." Unlike traditional software bugs that produce obviously wrong outputs (null values, exceptions, system crashes), hallucinations appear plausible. They arrive with the same presentation and the same confidence indicators as correct outputs.

Consider an AI system predicting cabin temperature for a crewed vehicle. A traditional software fault might produce a null value or an exception, clearly wrong, immediately flagged. A hallucination might predict a cabin temperature of -45°C when the actual temperature is 21°C. The prediction is a valid floating-point number within the data type's range. It might even fall within historical bounds for certain mission profiles. But it is completely fabricated, unsupported by sensor data, contradicted by physical constraints, and dangerous if acted upon. Conventional error handling, designed to catch null values and format errors, would pass this output through without question.

The hallucination problem is context-dependent. A predicted temperature of -270°C would be valid near absolute zero but a hallucination for cabin temperature. This context-dependence means hallucination detection cannot be a generic filter; it must be grounded in domain-specific operational constraints and physical reality.

WHAT THE FRAMEWORK REQUIRES

The framework addresses hallucination through a five-stage defense: define, detect, respond, log, and independently validate.

Criteria definition requires each AI function to specify what constitutes a hallucination in its operational context, including outputs outside expected bounds, contradictory outputs, and outputs unsupported by inputs. Ground truth baselines must be established using authoritative data sources, physical constraints and conservation laws, cross-validation with independent sensors, and operator-verified reference datasets. Clear examples distinguishing hallucinations from valid edge-case outputs ensure consistent application.

Detection requires automated mechanisms implemented at inference time, not during post-mission analysis, when the damage is already done. Detection methods may include consistency checks against physical constraints, cross-sensor comparison, confidence thresholding, or output trajectory analysis. Detection must be validated against known hallucination test cases, including injected violations of physical constraints, sensor contradictions, and out-of-range predictions.

Response ensures that detection leads to protection. Detected hallucinations must be flagged and quarantined, autonomous execution must be blocked for flagged outputs, and human review must be triggered. Overriding a hallucination flag requires documented rationale. Detection without response provides no protection.

Logging captures hallucination events with full context (timestamps, inputs, outputs, detection method, confidence, and human review decisions), enabling trend analysis to determine whether hallucination frequency is increasing and informing retraining decisions.

Independent output validation provides the deepest layer of defense for safety-critical and mission-critical functions. The AI system's outputs must be validated against an independent source prior to execution. Critically, this independent source must not share common failure modes with the primary AI: no common training data,

sensor inputs, processing hardware, or software dependencies. A single fault that defeats both the primary output and its validation provides no protection.

VERIFICATION APPROACH

Verification combines inspection and test. Inspection confirms criteria definition, ground truth baselines, and logging completeness. Testing injects hallucinated outputs (violations of physical constraints, sensor contradictions, fabricated values) and confirms that detection identifies them, response prevents autonomous action, and independent validation catches discrepancies. Analysis confirms that independent validation sources use dissimilar methods and share no common failure modes with the primary AI.

9. Out-of-Distribution Detection (AI-6)

THE FAILURE MODE

ML models are only reliable within the distribution of data they were trained on. This is a fundamental constraint, not a limitation that better training can fully overcome. When inputs differ significantly from training data (due to sensor anomalies, novel operational scenarios, or environmental conditions not anticipated during development), model outputs become unreliable and potentially dangerous.

The insidious aspect of this failure mode is that many ML models, particularly neural networks, produce high-confidence predictions on out-of-distribution inputs. A classifier trained on images of cats and dogs, when presented with an image of a truck, may confidently classify it as a dog rather than indicating uncertainty. This behavior is not a bug in any specific model; it is a fundamental property of how most ML architectures compute confidence. The model does not know what it does not know.

Traditional software has defined input/output bounds that can be checked explicitly. An input validation function can reject values outside acceptable ranges. AI systems are vulnerable to inputs that are technically within valid data type ranges but far from anything the model was trained to handle, and the model's own confidence score provides no reliable indication that this has occurred.

WHAT THE FRAMEWORK REQUIRES

The framework requires four capabilities: characterize what "in-distribution" looks like, detect when operational inputs fall outside that distribution, respond appropriately when OOD inputs are detected, and log events for analysis and improvement.

Training distribution characterization provides the baseline against which operational inputs are compared. Documentation must include statistical measures (mean, variance, covariance, range, density estimates) sufficient to support automated OOD detection at runtime. Without explicit characterization, OOD detection cannot be implemented systematically.

Runtime OOD detection must occur during operation, not just during development testing. Detection methods may include statistical distance measures such as Mahalanobis distance, density estimation, ensemble disagreement, or dedicated OOD detection models. Detection must meet operational timing requirements to enable timely response.

OOD response prevents autonomous action on out-of-distribution inputs. When an input is flagged as OOD, the system must either escalate to human review or invoke predefined fallback behavior appropriate to the AI function's criticality. The response should scale with system classification: Safety-Critical AI may require mandatory human review, while Operational Support AI may use automated fallback.

OOD event logging enables post-mission analysis to identify patterns, assess whether the operational environment has shifted from training assumptions, and inform decisions about model retraining or operational envelope expansion.

VERIFICATION APPROACH

Verification is primarily test-based. Tests inject known OOD inputs, both synthetic examples and realistic operational scenarios, and confirm that detection mechanisms identify them correctly, response procedures are triggered, and autonomous execution is blocked or fallback behavior is invoked. Inspection confirms distribution characterization and logging completeness. Analysis confirms that detection methods are appropriate for the model architecture and that characterization is traceable to training datasets.

10. Adversarial Robustness (AI-7)

THE FAILURE MODE

AI systems are vulnerable to attack vectors that do not exist for traditional software. Adversarial attacks exploit the mathematical properties of learned models to manipulate behavior in ways that are difficult to detect and potentially devastating in safety-critical applications.

The attack surface spans the entire AI lifecycle. Training data can be poisoned: malicious samples injected to create backdoors that trigger dangerous behavior under specific conditions. Runtime inputs can be crafted with perturbations imperceptible to human observers but sufficient to cause dramatic misclassification. Model parameters can be extracted through systematic querying, enabling attackers to study vulnerabilities offline. Sensitive training data can be reconstructed from model outputs through inversion attacks.

What makes adversarial attacks uniquely dangerous is their subtlety. A poisoned model may function normally on the vast majority of inputs, passing all standard testing, while harboring a backdoor that activates only under attacker-controlled conditions. An adversarial input may differ from a legitimate input by less than a fraction of a percent in pixel values while producing a completely different classification.

WHAT THE FRAMEWORK REQUIRES

The framework addresses adversarial robustness across six dimensions, covering the full attack surface.

Data poisoning protection requires authenticated data sources, validated data integrity, and anomaly detection capable of identifying statistically suspicious training samples. Poisoning attack scenarios, including backdoor triggers, label flipping, and gradient-based poisoning, must be tested against known patterns.

Adversarial input protection requires input validation, perturbation detection, and verification that model outputs are stable under small input changes. Known attack patterns (FGSM, PGD, C&W) must be tested and model response documented.

Model inversion and extraction protection requires access controls, query rate limiting, and output perturbation or differential privacy techniques appropriate to training data sensitivity.

Model integrity ensures the deployed model is the validated model. Cryptographic signing provides tamper-evident verification; integrity checks at load time and during operation detect unauthorized modifications.

Adversarial event logging captures all security events with timestamps, severity, detection confidence, and system response, in immutable logs retained for post-mission forensic analysis.

AI supply chain security extends integrity verification to external dependencies: ML frameworks, libraries, and pre-trained model components. Dependencies may contain vulnerabilities, backdoors, or undocumented behaviors introduced through compromised maintainers, malicious updates, or unknown training data in pre-trained models.

VERIFICATION APPROACH

Verification combines analysis, inspection, and test. Analysis confirms data source authentication, integrity checks, and supply chain risk assessments. Inspection confirms cryptographic signing, access controls, and logging completeness. Testing exercises the system against known adversarial attacks, poisoning scenarios, and input perturbations, confirming that protections detect and respond appropriately.

11. Explainability (AI-8)

THE FAILURE MODE

Traditional software has explicit decision logic that can be inspected through code review. An engineer can trace exactly why a particular output was produced: which branch was taken, which condition evaluated to true, what data was used. This traceability is foundational to safety assurance: if you cannot explain why the system did what it did, you cannot verify that it will do the right thing next time.

AI systems, particularly neural networks, make decisions through learned patterns in weights that are not directly interpretable. A neural network with millions of parameters does not contain a readable decision tree. Its "reasoning" is distributed across weight matrices in a form that no human can meaningfully inspect. This creates a dangerous gap: operators receive outputs and recommendations without the ability to assess whether the AI's reasoning is sound.

The result is a binary trust failure. Some operators develop blind trust, accepting AI outputs without scrutiny because they have no way to evaluate the reasoning. Others develop blanket distrust, ignoring valid AI recommendations because the system cannot explain itself. Neither response is appropriate for safety-critical operations.

WHAT THE FRAMEWORK REQUIRES

The framework requires explainability capabilities that scale with decision criticality and are designed for the operator's needs, not the engineer's convenience.

Explainability requirements definition ensures that explainability is designed in from the beginning, not retrofitted. Requirements must be scaled to function criticality (Safety-Critical AI functions require more rigorous explanation capabilities than Operational Support functions) and must consider the target audience. Operators need operationally relevant explanations within operational time constraints. Engineers may need deeper technical detail for post-mission analysis. The explainability approach must be justified for each AI function.

Decision explanation generation requires that explanations be actionable and comprehensible to operators. Technical explanations such as feature weights and activation patterns may be appropriate for post-mission analysis but are inadequate for real-time operational decisions. Explanation format must be validated through human factors assessment to ensure operators actually understand and can act on the information provided.

Confidence indication addresses the critical need for operators to know when to trust AI outputs. Confidence must be calibrated: a stated 90% confidence should be correct approximately 90% of the time. Many AI systems, particularly neural networks, produce poorly calibrated confidence scores. The framework requires calibration validation with an expected calibration error of 0.05 or less, and clear visual differentiation between high-confidence and low-confidence outputs.

Reasoning inspection capability provides on-demand access to deeper explanation for safety-critical decisions. When an AI recommendation conflicts with operator expectations or when uncertainty exists, operators must be able to drill down into the AI's reasoning without unacceptable delay to operational timelines.

VERIFICATION APPROACH

Verification is primarily test-based, with inspection for requirements documentation. Testing confirms that explanations are generated, that format is appropriate to context, that key influencing factors are identified, that confidence calibration meets required thresholds, and, critically, that operator comprehension is validated

through usability assessment with representative operators performing representative tasks. Explainability that operators cannot understand provides no protection.

12. Human-AI Teaming (AI-9)

THE FAILURE MODE

AI systems in safety-critical operations are part of a human-machine team. They are not fully autonomous agents operating independently. The failure mode this requirement area addresses is not a failure of the AI alone; it is a failure of the team. Automation complacency causes operators to stop checking AI outputs. Trust miscalibration leads to either blind acceptance or wholesale rejection of valid recommendations. Loss of situational awareness leaves operators unable to intervene when intervention is needed. And when AI capabilities degrade, operators who have been relying on AI support may find themselves unable to continue operations manually.

These are human factors failures, but they are caused by system design. An AI system that provides no visibility into its state invites loss of situational awareness. An AI system that presents all outputs with equal formatting and no uncertainty indication invites overtrust. An AI system that fails abruptly rather than degrading gracefully leaves operators without support and without warning.

WHAT THE FRAMEWORK REQUIRES

The framework requires five capabilities that collectively ensure effective human-AI collaboration.

Human decision authority is the foundational requirement: for safety-critical actions, humans retain final decision authority. AI may recommend, advise, or prepare actions, but execution requires human authorization except where explicitly designed, approved, and documented for autonomous operation. Human override capability must always be available and functional.

Operational situational awareness requires that the AI's operational state be visible to operators through appropriate displays. Changes in AI state (mode transitions, confidence drops, detected anomalies) must be communicated clearly and within operational timing constraints. A state change notification latency of 500 milliseconds or less and a confidence drop greater than 15% triggering an alert are representative requirements, aligned with human factors reaction time research.

Trust calibration addresses the dual failure of overtrust and undertrust. The system must document AI reliability and limitations, communicate them to operators, and include training materials addressing scenarios where AI should and should not be trusted. System design must discourage both overtrust (through uncertainty indication and limitation warnings) and undertrust (through reliability demonstration and calibrated confidence). Trust calibration must be assessed through human factors evaluation with representative operators.

Graceful degradation ensures that when AI capabilities are reduced (whether from sensor failures, computational constraints, or detected anomalies), operators are notified and manual or backup procedures are available. Transition to degraded modes must be achievable in two or fewer operator actions and five seconds or less, validated under time pressure.

Human-AI interaction logging captures what the AI recommended, what the human decided, and the rationale for any overrides. This data supports post-mission analysis of trust calibration, override patterns, and human factors issues, enabling continuous improvement of human-AI teaming effectiveness.

VERIFICATION APPROACH

Verification is primarily test-based, with inspection for logging. Testing confirms that human override functions correctly, that state changes are annunciated within timing requirements, that trust calibration mechanisms function, that degraded modes are achievable, and that operators can determine what the AI is doing and why.

Human factors evaluation with representative operators is required for trust calibration assessment, a verification method not typically found in traditional software standards, but essential for systems where the human–AI interface is a safety-critical boundary.

13. Privacy and Data Protection (AI-10)

THE FAILURE MODE

Traditional software treats privacy as a procedural matter, a policy layer that lives outside the system in consent forms, access controls, and retention schedules. For AI systems that learn from personal data, that assumption fails structurally: the model itself becomes a privacy-relevant artifact. Training data may be reconstructable from model parameters: model inversion attacks recover sensitive training samples by systematically analyzing model outputs. Trained models may leak membership information, allowing an adversary to determine whether a specific individual's data was used in training. And the opacity of AI decision-making complicates the exercise of legally mandated data subject rights. Erasure may require retraining when personal data has shaped learned parameters. Rectification of training data does not retroactively correct trained behavior. Access rights may require explaining an automated decision the system cannot natively explain.

Left unaddressed, these failure modes produce two distinct harms: regulatory violation under privacy frameworks such as GDPR, HIPAA, and CCPA/CPRA, and genuine privacy compromise of the individuals whose data trained the system. Neither is hypothetical; model inversion and membership inference are demonstrated attack classes, not theoretical concerns.

WHAT THE FRAMEWORK REQUIRES

The framework requires seven capabilities, scoped by a single applicability rule: AI-10 applies wherever the system processes personal data under any applicable privacy framework. Where it does not, sub-requirements are documented as Not Applicable with rationale, a determination recorded in the operational design domain declaration (AI-1.0).

Personal data identification and minimization requires identifying personal data across all training, validation, test, and operational datasets, and limiting collection, retention, and processing to what the declared purpose requires.

Lawful basis and consent management requires documenting the lawful basis for processing under each applicable privacy framework, and implementing consent management mechanisms where consent is the basis.

Data subject rights implementation requires procedures supporting access, rectification, erasure, restriction, portability, and objection, with the AI-specific complications addressed directly. Erasure requests trigger an assessment of whether model retraining is required. Explainability outputs (AI-8) serve as supporting evidence when access requests require explanation of automated decisions.

Privacy-enhancing techniques must be applied in proportion to data sensitivity, processing purpose, and re-identification risk: anonymization, pseudonymization, differential privacy, federated learning, or secure aggregation as appropriate. The framework treats these techniques as non-interchangeable: an approach that satisfies one regulatory regime's anonymization standard may not satisfy another's.

Cross-border transfer controls address the transfer points unique to AI systems: cloud-hosted training infrastructure, third-party annotation services, regional inference endpoints, and model weights derived from training data that crossed jurisdictions.

The privacy impact assessment is the integrating artifact: a PIA or DPIA, conducted where regulation requires it, that ties the other capabilities together and documents privacy risk mitigation across the AI lifecycle.

The privacy compliance audit trail captures privacy-relevant events (rights requests and their dispositions, consent grants and withdrawals, transfer authorizations, privacy incidents, and PIA reviews) as a privacy-domain complement to the data audit trail in AI-1. The two trails serve distinct regulatory regimes and are maintained separately even where logging infrastructure is shared.

VERIFICATION APPROACH

Verification combines inspection, analysis, and test. Testing must demonstrate that representative data subject rights requests can be processed end-to-end within regulatory timelines (thirty days under GDPR and HIPAA, forty-five under CCPA) and that erasure requests produce documented dataset removal and a retraining-necessity assessment. Analysis must validate anonymization claims against the applicable regulatory standard rather than in the abstract, and document differential privacy parameters with a privacy budget and composition analysis where differential privacy is claimed. The privacy impact assessment must be approved by an authority with privacy responsibility, a Data Protection Officer where required or a designated equivalent. Evidence artifacts include the personal data identification and minimization record, lawful basis and consent records, rights-request response evidence, the privacy-enhancing technique selection and validation report, the data flow map with transfer mechanisms, the PIA record, and the privacy event audit trail.

PART IV

Architecture and Paradigm Requirements

The ten core requirement areas are deliberately algorithm-agnostic: they apply whether a system is built on neural networks, decision trees, ensembles, or foundation models. But design choices carry failure modes of their own. A perception model feeding a planning model can fail in ways neither model exhibits alone. A neural network fails in ways a decision tree cannot. A system that keeps learning after deployment fails in ways a frozen model never will.

Version 3.0 addresses this with a second requirement tier. Unlike the core, which applies to every AI system, the three requirement areas in this part apply conditionally, when the architecture or paradigm that produces their failure modes is present. The algorithm-agnostic baseline stays broadly applicable; paradigm-specific failure modes still receive normative coverage. Tailoring follows the same classification-scaled authority as the core requirements.

14. Multi-Model Systems (AI-11)

THE FAILURE MODE

Modern systems increasingly incorporate multiple AI models operating in concert: a perception model feeding a planning model, multiple specialized classifiers contributing to a fused decision, retrieval-augmented generation pipelines, ensembles aggregated by voting. These architectures introduce failure modes not present in single-model systems, and a system can exhibit them even when every constituent model performs within its own specification.

Error propagation through cascaded models can amplify or mask upstream errors. Combined confidence across a model chain is generally lower than any individual model's confidence, and misrepresenting it overstates the system's certainty. Out-of-distribution outputs from upstream models may appear in-distribution to downstream models, silently defeating OOD detection. Sequential processing accumulates latency that may breach time-critical decision windows. And for ensemble and voting systems, correlated failures (models trained on similar data failing on the same inputs) undermine the independence assumption that makes voting effective.

WHAT THE FRAMEWORK REQUIRES

The framework requires five capabilities for any system that composes two or more AI models.

Multi-model architecture documentation establishes the analytical foundation: the role of each model, the data flows between models, model-to-model interface specifications, combined system-level performance requirements, and a failure mode analysis that explicitly addresses cascading and correlated failure scenarios.

Cascading failure mitigation requires implemented, documented mitigations for chained architectures: error propagation between sequential models, confidence degradation across the chain, out-of-distribution propagation, and latency accumulation in time-critical decision paths.

Ensemble coordination requires controls against correlated failures, a documented rationale for voting thresholds (majority, unanimous, or weighted), and defined disagreement handling when models diverge beyond thresholds.

Combined confidence representation requires that displayed confidence accurately represent the aggregate uncertainty across all participating models: combined system-level confidence clearly labeled, individual model contributions visible on demand, visual indication when models disagree beyond defined thresholds, and a clear mapping from confidence level to recommended operator action.

The multi-model audit trail records which models contributed to each output, their contribution weights or confidence values, the aggregation method applied, and the resulting decision rationale, retained under the same rules as the framework's other audit trails.

VERIFICATION APPROACH

Architecture documentation and the audit trail are verified by inspection. Cascading failure mitigation and ensemble coordination are verified by analysis and test, including tests that inject upstream errors and out-of-distribution inputs to confirm they are detected or contained rather than silently propagated. Combined confidence representation is verified by test against the documented aggregation method. Evidence artifacts include the architecture documentation, cascade and ensemble test results, confidence display test results, and the multi-model decision audit trail.

15. Neural Networks (AI-12)

THE FAILURE MODE

Neural networks are the dominant architecture in modern AI, and they carry a distinctive set of failure modes. There is no inspectable decision logic: post-hoc explanation methods such as SHAP, LIME, and attention visualization produce approximations that may not reflect true model reasoning. Confidence is routinely miscalibrated: a model reporting 90% confidence may be correct only 70% of the time, and softmax outputs tend to remain high even on inputs far outside the training distribution. And networks with sufficient capacity can memorize training data rather than learning generalizable patterns.

Two further properties demand attention. Neural networks are uniquely susceptible to adversarial examples (a fundamental property of gradient-based learning that cannot be fully eliminated), and some published defenses create a false sense of security through gradient masking rather than genuine robustness improvement. Neural networks are also particularly sensitive to deployment transformations: quantization, pruning, distillation, and framework conversion can each alter behavior in ways aggregate metrics do not reveal. Where the network generates outputs (text, trajectories, plans, code), hallucination joins the list.

WHAT THE FRAMEWORK REQUIRES

The framework requires eight capabilities for any system incorporating neural network components.

Architecture documentation goes beyond standard software documentation: layer types, dimensions, and connectivity; activation functions and normalization schemes; parameter count and memory footprint; computational requirements for inference; optimizer configuration and training schedule; data augmentation techniques; hardware and framework versions; accuracy across data subsets and operational scenarios; calibration curves for confidence outputs; latency on target hardware; known failure modes and limitations; and the deployment artifact chain with cryptographic hashes at every transformation step.

Confidence calibration requires demonstrating that confidence outputs are calibrated against actual prediction accuracy, applying temperature scaling, Platt scaling, or isotonic regression where calibration error exceeds documented thresholds, and validating calibration on representative held-out data.

Explainability method selection must be documented and justified for the architecture and decision criticality. For Safety-Critical functions, projects must document the rationale for selecting a neural network over inherently interpretable alternatives (decision trees, rule-based systems, linear models) when an interpretable model could achieve the required performance.

Training integrity requires documented overfitting indicators and the threshold beyond which retraining is required, a documented relationship between model complexity and training data volume, documented random seeds and training configurations for reproducibility, and validation-set contamination controls that specifically address hyperparameter tuning and architecture decisions informed by validation performance.

Out-of-distribution detection extensions recognize that softmax confidence alone is insufficient. Detection must use methods that account for the tendency of neural networks to produce high-confidence predictions far from the training distribution: ensemble disagreement, energy-based methods, feature-space distance metrics, or deep generative model likelihood.

Adversarial vulnerability mitigation requires testing against gradient-based attack classes (FGSM, PGD, C&W) and gradient-free attacks, and explicitly evaluating defenses for gradient masking, where apparent robustness improvements do not survive attacks that bypass the gradient.

Generative hallucination constraints apply to networks that generate outputs: physical plausibility checks against conservation laws, kinematic limits, and domain-specific bounds; cross-validation with independent sensors or models where available; confidence-bounded outputs that refuse to generate beyond validated thresholds; and output format constraints that limit generable outputs to verifiable forms.

Deployment transformation validation extends the deployment format validation of Section 18.3 with neural-network-specific rigor: per-layer quantization sensitivity analysis with post-quantization calibration on representative data, validation that pruned models do not exhibit different failure modes than the original, independent validation of distilled student models on out-of-distribution inputs and edge cases, and numerical equivalence validation across framework conversions with documented acceptable tolerances.

VERIFICATION APPROACH

Documentation and training integrity are verified by inspection and analysis. Calibration is verified by analysis and test against held-out data. OOD detection extensions, adversarial mitigations, and transformation validation are verified by test, including adversarial campaigns designed specifically to detect gradient masking. Hallucination constraints are verified by analysis and test against the documented constraint set. The evidence package mirrors the requirement structure: neural network architecture documentation, calibration reports, OOD and adversarial test results, transformation equivalence reports, and hallucination constraint validation.

16. Continuous Learning and Adaptation (AI-13)

THE FAILURE MODE

Most certified AI systems are frozen: trained, validated, deployed, and unchanged until the next release. Continuous learning systems update their model parameters during operation, performing incremental training, as ISO/IEC 22989 defines it, while the system runs. The capability is attractive precisely where environments are dynamic, and it can directly counter the drift failure modes addressed in AI-4. But it introduces failure modes that statically trained systems do not exhibit.

Catastrophic forgetting is the headline risk: new learning overwrites previously validated capabilities, and the behavior that passed certification quietly disappears. Learning from operational data means learning from whatever the operational environment supplies, including drifted, anomalous, or deliberately poisoned data. Online learning can become unstable. And underneath all of these sits the deepest verification problem: a system that changes during operation cannot be verified once against a fixed validation baseline, because the model that was verified is no longer the model that is running.

WHAT THE FRAMEWORK REQUIRES

Permission criteria come first. The system must declare the conditions under which continuous learning is permitted: the classification tier, the operational scenarios in which learning may occur, the data sources from which learning may proceed, and the controls required at each tier. The controls scale with criticality. Safety-Critical AI requires rollback capability, approval of model updates prior to activation, shadow-mode validation before operational use, and automated performance-bounds monitoring. Mission-Critical AI requires performance monitoring, automated rollback at defined thresholds, logging of all model updates, and periodic review of the learning trajectory. Operational Support AI requires baseline performance tracking and anomaly alerting for significant behavior changes.

Runtime learning controls prevent learning when defined preconditions are not met: during safety-critical operational windows, when learning data sources are suspect, or when system confidence is below validated thresholds.

Learning data validation screens what the system learns from: data freshness relative to the operational distribution, source legitimacy, and anomaly detection for poisoning indicators before data is incorporated into any update.

Catastrophic forgetting mitigation must be implemented and verified: elastic weight consolidation, progressive neural networks, replay buffers, or other techniques appropriate to the model architecture and learning paradigm.

Learning validation and rollback close the loop. Updates are validated against benchmark performance criteria before the updated model becomes the operational model; updates that fail validation are rejected and the prior validated model retained. When a degraded update reaches operation despite these gates, rollback restores a prior validated model state, invokable both automatically (on threshold-triggered detection) and manually (by qualified operator authority).

ODD evolvability must be declared. The system declares whether its operational design domain is permitted to evolve through continuous learning, the bounds within which evolution may occur, and the conditions that trigger ODD reassessment. A continuous learning audit trail records every learning event: the data sources used, the parameter updates applied, validation outcomes, and any rollback actions taken.

Classification interacts with this requirement area directly: a system that learns during operation meets the potential-drift criterion and is classified no lower than Mission-Critical.

VERIFICATION APPROACH

Verification is test-heavy, as it should be for a paradigm whose risks are behavioral. Testing must demonstrate that runtime controls actually prevent learning under blocked preconditions, that forgetting mitigations preserve previously validated capabilities, that updates failing validation are rejected, and that rollback works, automatically and manually, under realistic conditions. Permission criteria, the ODD evolvability declaration, and the audit trail are verified by inspection. Evidence artifacts include the permission criteria documentation, learning data validation records, forgetting mitigation test results, update validation records, rollback test results, and the continuous learning audit trail.

PART V

Verification and Implementation

17. Operational Threshold Categories: The Enforcement Mechanism

Every requirement in this framework (from bias detection to hallucination prevention to graceful degradation) depends on a common mechanism: defined thresholds that convert abstract protections into concrete, verifiable, pass/fail criteria. Without thresholds, "monitor for data drift" is an intention. With thresholds, it becomes "alert when PSI exceeds 0.1; halt autonomous operation when PSI exceeds 0.25." The difference is the difference between a safety standard and a suggestion.

For this reason, threshold documentation is not optional and is not guidance. All AI systems, regardless of classification, must document thresholds across five categories:

Category	What It Governs	Without It
Performance Degradation	Minimum acceptable accuracy, precision, recall, or domain-appropriate metric below which system response is required	No defined point at which degraded performance triggers action; system operates with declining reliability until failure
Confidence	Minimum output confidence for autonomous action versus human review escalation	System acts on low-confidence outputs with the same authority as high-confidence outputs; operators receive no signal to apply scrutiny
Data Drift	Maximum deviation from training distribution before revalidation is triggered	Distribution shift goes undetected; model operates on data it was never validated against
OOD Detection	Boundary conditions classifying inputs as out-of-distribution, requiring fallback behavior	System produces predictions on inputs far outside training data with no mechanism to recognize or respond
Human Escalation	Conditions requiring mandatory human review before action, independent of other thresholds	No defined conditions for mandatory human oversight; all decisions default to autonomous execution

These five categories are necessary for adequate operational safety. They are not exhaustive, additional categories may be needed based on specific AI function characteristics and mission context. But these five represent the minimum set without which the framework's protections cannot be operationally enforced.

Threshold values are project-defined. The framework does not prescribe that a data drift PSI threshold must be 0.25, or that confidence must exceed 0.85 for autonomous action. These values depend on mission risk tolerance, operational context, and the specific AI function. What the framework does prescribe is that values must exist, must be documented with derivation rationale, must have defined system responses when exceeded, and must be validated as appropriate through analysis or test.

Threshold documentation must capture: the selected value and units, the derivation basis (analysis, heritage data, industry standard, simulation), the relationship to mission risk tolerance, the system response when the threshold is exceeded, validation evidence demonstrating the threshold is appropriate, and conditions under which the threshold should be revisited.

Threshold stringency scales with classification. Safety-Critical AI requires the most stringent thresholds with conservative margins, derived from hazard analysis and safety margins. Mission-Critical AI balances thresholds against mission objectives. Operational Support AI may apply relaxed thresholds where operational impact is limited, with emphasis on trend detection over absolute boundaries.

18. Verification Architecture

18.1 Verification Methods

Verification of AI-specific requirements uses four standard methods from IEEE 1012, tailored for AI/ML system characteristics. Inspection examines documentation, artifacts, and code, applied to dataset documentation, provenance records, and log schemas. Analysis uses mathematical or analytical techniques for bias risk assessment, coverage analysis, and distribution characterization. Demonstration observes system behavior under realistic conditions for operator comprehension assessment and human-AI interaction evaluation. Test executes the system under controlled conditions with pass/fail criteria for drift detection, OOD detection, hallucination detection, and adversarial testing.

The balance of methods shifts significantly from traditional software verification. Traditional verification is heavily weighted toward inspection (code review) and test (functional testing). AI verification relies more heavily on analysis (statistical methods, distribution characterization) and demonstration (human factors evaluation), while still using inspection and test extensively.

18.2 Verification Matrix Summary

The following table summarizes the primary verification methods and key evidence artifacts for each requirement area.

Area	Focus	Primary Methods	Key Evidence Artifacts
AI-1	Data Integrity	Inspection, Analysis	Partition records, provenance docs, labeling reports
AI-2	Bias	Insp., Analysis, Test	Bias baselines, monitoring logs, impact assessments, edge case results
AI-3	Test Coverage	Insp., Analysis, Test	Coverage metrics, threshold reports, distribution test results
AI-4	Continuous Valid.	Insp., Analysis, Test	Drift logs, performance trends, boundary test results
AI-5	Hallucination	Insp., Analysis, Test	Criteria docs, detection reports, independent validation evidence
AI-6	OOD Detection	Insp., Analysis, Test	Distribution characterization, OOD reports, event logs
AI-7	Adversarial	Insp., Analysis, Test	Adversarial test reports, integrity logs, supply chain assessment
AI-8	Explainability	Inspection, Test	Explainability assessment, calibration report, usability results
AI-9	Human-AI Team	Inspection, Test	Authority tests, SA tests, degradation tests, interaction logs
AI-10	Privacy	Insp., Analysis, Test	PII identification records, lawful basis and consent records, DSR response evidence, PIA/DPIA, privacy audit trail
AI-11	Multi-Model	Insp., Analysis, Test	Architecture documentation, cascade and ensemble test results, confidence display tests, decision audit trail
AI-12	Neural Networks	Insp., Analysis, Test	NN architecture docs, calibration report, OOD and adversarial test results, transformation equivalence reports

AI-13	Continuous Learning	Inspection, Test	Permission criteria, learning validation records, rollback test results, learning audit trail

Table 4. Verification matrix summary.

For the architecture and paradigm requirement areas (AI-11 through AI-13), verification statements are co-located with each sub-requirement, each carrying its own success criteria; their audit trails follow the same retention rules as the core areas.

18.3 Deployment Format Validation

When AI models are transformed for deployment on edge compute systems, the deployment-format model must be revalidated against the original trained model. These transformations are common and can significantly alter model behavior.

Quantization reduces model precision, for example, converting from 32-bit floating point to 8-bit integer representation. This reduces memory and compute requirements but can cause catastrophic accuracy degradation in some layers while having minimal impact on others. Layer-wise sensitivity analysis should identify critical layers requiring higher precision, and post-quantization calibration using representative data can recover lost accuracy.

Pruning removes model weights or layers to reduce size. Pruned models may exhibit different failure modes than original models even when aggregate performance metrics are similar, a particularly dangerous situation that requires validation against edge cases and adversarial inputs, not just average-case performance.

Knowledge distillation trains a smaller student model to mimic a larger teacher model. The student may not preserve the teacher's behavior on out-of-distribution inputs or edge cases, and must be validated independently against all requirements. Inherited verification from the teacher model is insufficient.

Framework conversion between different ML frameworks (for example, PyTorch to ONNX to TensorRT) can introduce numerical differences due to operator implementation variations. Acceptable numerical tolerance thresholds must be documented, and equivalence validated with representative inputs.

In all cases, revalidation must demonstrate that the deployment-format model meets all performance thresholds established for the original model. Performance degradation due to transformation must be documented and accepted within defined margins.

18.4 Verification Evidence Architecture

A regulatory framework is only as credible as the evidence it produces. Each requirement in this framework generates specific, auditable verification evidence that can be reviewed by independent assessors, retained for program records, and used in safety case argumentation.

Verification evidence falls into three categories corresponding to three questions an auditor would ask:

Does AI apply, and how is it classified? The AI Applicability Determination Record documents whether the system meets AI criteria per Section 3.2, the resulting classification (Safety-Critical, Mission-Critical, or Operational Support), applicable requirement areas, any tailoring decisions with rationale, and approval authority. This record is completed once per system or subsystem and provides the foundation for all subsequent verification.

Has each requirement been verified? For each applicable sub-requirement, a verification record documents the verification method used (inspection, analysis, demonstration, or test), the specific evidence examined or produced, the success criteria and whether they were met, any findings or non-conformances identified, and the compliance determination. For test-based verification, this includes test environment, test conductor, pass/fail criteria, results, and anomaly disposition. For inspection-based verification, this includes the artifact inspected, inspection criteria, and findings.

Is the evidence traceable and complete? The verification matrix traces every sub-requirement to its verification method, responsible party, evidence artifact, and completion status. This matrix serves as the master index for the verification evidence package and enables rapid assessment of verification completeness.

The evidence package for a Safety-Critical AI system is substantial. It includes dataset partition records with provenance documentation, bias baseline reports and continuous monitoring logs, test coverage metrics and threshold achievement reports, drift detection logs and periodic validation results, hallucination criteria documentation and detection test reports, OOD distribution characterization and detection evidence, adversarial test reports and model integrity verification logs, explainability assessment and confidence calibration reports, human-AI teaming test results and interaction logs, and threshold documentation with derivation rationale for all five mandatory categories.

This evidence is not paperwork for its own sake. Each artifact exists because it demonstrates that a specific protection against a specific failure mode is in place and functioning. An auditor reviewing the evidence package can determine, for each documented AI failure mode, whether the system has verified protections, and where it does not.

19. Implementation Considerations

Earlier versions of this white paper carried implementation guidance for continuous learning, multi-model systems, and neural networks in this part. Version 3.0 promotes that content to normative requirement areas: AI-13, AI-11, and AI-12 respectively, covered in Part IV. What remains here is guidance that is genuinely advisory: how to select threshold values, and how to integrate AI-specific failure modes into hazard analysis.

19.1 Threshold Selection Guidance

The framework requires projects to define and document thresholds for bias metrics, drift detection, performance monitoring, and other AI-specific measures. The following table provides example threshold ranges and selection considerations representing typical industry practice. These examples are informative and should be tailored based on mission risk tolerance, operational context, and system classification.

Area	Metric	Example Range	Selection Considerations
AI-2 (Bias)	Demographic parity	≤2% safety-crit; ≤5% mission-crit	Tighter for crew health/safety; consider sample sizes
AI-2 (Bias)	Equalized odds diff.	≤3% TPR/FPR across groups	Critical for detection/diagnostic systems
AI-4 (Drift)	Data drift (PSI)	<0.1 neg; 0.1-0.25 mod; >0.25 sig	Alert at moderate; halt autonomous at significant
AI-4 (Drift)	KL divergence	>0.1 investigate; >0.2 action	Monitor per feature; aggregate may mask localized drift
AI-5 (Halluc.)	Detection rate	≥95% detect; ≤1% false positive	Balance sensitivity against operational false alarm burden
AI-3 (Coverage)	Neuron activation	≥95% safety-crit; ≥90% mission-crit	Complement with input space coverage
AI-6 (OOD)	OOD detection rate	≥97% detect; ≤2% false positive	Validate against realistic OOD scenarios
AI-8 (Explain.)	Confidence cal. (ECE)	≤0.05; slope 0.9-1.1	Critical for operator trust calibration
AI-9 (Human)	State change notif.	≤500ms; >15% conf drop alerts	Align with human factors reaction time
AI-9 (Human)	Manual mode transition	≤2 actions; ≤5 seconds	Validate under time pressure in training

Table 5. Example threshold ranges (informative).

Threshold stringency should scale with classification. Safety-Critical AI requires the most stringent thresholds with conservative margins and multiple independent detection mechanisms. Mission-Critical AI balances thresholds against mission objectives. Operational Support AI may relax thresholds where operational impact is limited, focusing on trend detection over absolute thresholds.

19.2 AI-Specific Hazard Analysis Integration

Traditional hazard analysis methods (FMEA, FTA, HAZOP) assume deterministic failure modes. AI systems introduce probabilistic, emergent, and adversarial failure modes that require augmented analysis. When conducting hazard analysis for AI systems, the analysis should include data-related failures (bias, drift, poisoning, leakage), model-related failures (hallucination, overconfidence, underfitting, overfitting), operational failures (OOD

inputs, concept drift, degradation), adversarial failures (input manipulation, model theft, poisoning attacks), and human factors failures (automation bias, overtrust, undertrust).

PART VI

Standards Alignment and Conclusion

20. Normative References and Standards Crosswalk

The framework supplements, and is designed for integration with, established safety-critical software standards. Version 3.0 designates ten normative references: ISO/IEC 22989:2022 (AI concepts and terminology), ISO/IEC 5338:2023 (AI lifecycle processes), NIST AI RMF 1.0 (risk taxonomy), DO-178C (aviation software), ISO 26262 (automotive functional safety), ISO/IEC 27701:2019 (privacy information management), ISO/IEC 29100:2011 (privacy framework), the NIST Privacy Framework v1.0, the EU AI Act (Regulation 2024/1689), and IEEE 1012-2016 (verification and validation methods). Informative references include NPR 7150.2D, NASA-STD-8739.8, ISO/IEC 23053:2022, ISO/IEC TR 24027:2021, NIST AI 600-1, the FAA AI Safety Roadmap, GDPR, the HIPAA Privacy and Security Rules, CCPA/CPRA, NIST SP 800-122, and ISO 34503.

Beyond the reference lists, every requirement carries inline citations to applicable domain standards (aviation, automotive, medical, aerospace, industrial, and privacy regimes) organized by domain. These inline citations are informative rather than normative: certification is conducted against the framework's own requirements, while the citations support cross-regulatory recognition where a single system must satisfy multiple regimes. Each citation is classified under the five-category source authority typology described in Appendix D, and citations that could not be verified at clause level against source documents are explicitly disclosed there.

The framework maps cleanly to ISO/IEC 5338:2023 lifecycle processes. AI system quality management maps to AI-3, AI-4, and the verification architecture. AI system risk management maps to the gap analysis, AI-2.5 (bias impact assessment), and hazard analysis guidance. AI data engineering maps to AI-1 (operational and data foundations). Continuous validation maps to AI-4, and model monitoring maps to AI-4 and AI-6. Verification processes map to AI-1 through AI-13 through the verification matrix. A complete crosswalk is maintained as a companion reference document.

21. Conclusion

Safety-critical AI systems need requirements designed around how they fail, not how they succeed. Current safety-critical software standards contain structural assumptions that do not hold for AI/ML systems. These assumptions create predictable gaps in safety assurance that leave documented AI failure modes unaddressed.

The AI Requirements Framework for Safety-Critical Systems addresses these gaps through thirteen requirement areas organized into two tiers: ten core areas that apply to every AI system, and three architecture- and paradigm-specific areas that apply when a system employs multi-model coordination, neural networks, or continuous learning. Each is defined by the failure it prevents and verified through standard methods with defined evidence artifacts. The framework supplements existing standards, integrates into existing software lifecycles, and scales with system classification from operational support through safety-critical applications.

Earlier versions of this framework acknowledged what they did not yet cover: multi-model cascading systems, continual learning architectures, and generative AI. Version 3.0 addresses each one, with AI-11 for multi-model coordination, AI-13 for systems that learn during operation, and AI-12's generative constraints for hallucination in generated outputs. The framework will continue to add architecture and paradigm requirement sets as the field matures. The core principle endures: safety-critical AI systems must be governed by standards that assume they will fail, that define how they must fail (gracefully, transparently, recoverably), and that verify, with auditable evidence, that those protections are in place.

The gap between where AI is being deployed and where the standards are is widening. This framework is one contribution toward closing it.

APPENDICES

Reference Material

Appendix A: Glossary

Definitions aligned with ISO/IEC 22989:2022 and ISO/IEC 5338:2023 where applicable. Framework-specific terms are noted.

Core AI/ML Terms

Artificial Intelligence (AI): Research and development of mechanisms and applications of AI systems. Computer systems designed to exhibit human-like learning, reasoning, judgement, and rational behavior. (ISO/IEC 22989:2022 3.1.3, tailored)

AI System: Engineered system that generates outputs such as content, forecasts, recommendations or decisions for a given set of human-defined objectives. (ISO/IEC 22989:2022 3.1.4)

Machine Learning: Process of optimizing model parameters through computational techniques, such that the model's behavior reflects the data or experience. (ISO/IEC 22989:2022 3.3.5)

Machine Learning Model: Mathematical construct that generates an inference or prediction based on input data. (ISO/IEC 22989:2022 3.3.7)

Continuous Learning: Incremental training of an AI system during the operation phase of the AI system life cycle. (ISO/IEC 22989:2022 3.1.9)

Human-Machine Teaming: Integration of human interaction with machine intelligence capabilities. (ISO/IEC 22989:2022 3.3.3)

Data Terms

Training Data: Data used to train a machine learning model. (ISO/IEC 22989:2022 3.3.16)

Validation Data: Data used to compare the performance of different candidate models. (ISO/IEC 22989:2022 3.2.15)

Test Data: Data used to assess the performance of a final model. (ISO/IEC 22989:2022 3.2.14)

Ground Truth: Value of the target variable for a particular item of labelled input data. (ISO/IEC 22989:2022 3.2.7)

Risk and Failure Mode Terms

Data Drift: Change in the statistical characteristics of production data over time compared to training data distribution. (ISO/IEC 5338:2023 6.4.14, tailored)

Concept Drift: Change in the relationship between input data and the target variable over time. (ISO/IEC 5338:2023 6.4.14, tailored)

Bias: Systematic difference in treatment of certain objects, people, or groups in comparison to others. (ISO/IEC 22989:2022 3.5.4)

Hallucination: The production of confidently stated but erroneous or false content. (NIST AI 600-1 Section 2.4)

Out-of-Distribution: Input data outside the statistical distribution of training data, for which predictions may be unreliable. (ISO/IEC 23053:2022, tailored)

Overfitting: Creating a model which fits training data too precisely and fails to generalize. (ISO/IEC 23053:2022 3.2)

Explainability and Trust Terms

Explainability: Property of an AI system to express important factors influencing results in a way that humans can understand. (ISO/IEC 22989:2022 3.5.7)

Trust Calibration: Alignment between operator trust in AI and actual AI reliability. (Human factors)

Confidence: Quantified measure of certainty associated with an AI output or prediction. (Tailored)

Graceful Degradation: System capability to maintain reduced functionality when AI is impaired. (Tailored)

Controllability: Property allowing human or external agent to intervene in functioning. (ISO/IEC 22989:2022 3.5.6)

Security Terms

Adversarial Attack: Deliberate manipulation of AI inputs, training data, or parameters to cause incorrect or harmful outputs. (ISO/IEC 5338:2023 6.4.3)

Data Poisoning: Injection of malicious data into training datasets to influence model behavior. (ISO/IEC 5338:2023 6.4.3)

Model Inversion: Attack reconstructing sensitive training data by analyzing model outputs. (ISO/IEC 5338:2023 6.4.3)

Privacy Terms

Personal Data: Information relating to an identified or identifiable natural person. (GDPR Article 4; ISO/IEC 29100:2011)

Membership Inference: Attack determining whether a specific individual's data was included in a model's training set.

Differential Privacy: A mathematical framework bounding how much any single training record can influence a model's outputs, quantified by a privacy budget.

Data Subject Rights: Legally mandated rights of individuals over their personal data, including access, rectification, erasure, restriction, portability, and objection. (GDPR Articles 15-21)

Privacy Impact Assessment: Structured assessment of privacy risks and mitigations across the AI lifecycle; a Data Protection Impact Assessment where GDPR applies. (GDPR Article 35)

Operational Terms

Fallback Behavior: Predefined safe system response when AI outputs are unreliable or unavailable. (Tailored)

Human Override: Capability for operators to supersede AI recommendations or autonomous actions. (Tailored)

Robustness: Ability to maintain performance under any circumstances. (ISO/IEC 22989:2022 3.5.12)

Architecture and Adaptation Terms

Continuous Learning: Incremental training that occurs during operation. (ISO/IEC 22989:2022 5.11.9.2)

Catastrophic Forgetting: Loss of previously validated capabilities when new learning overwrites earlier learning. (ISO/IEC 22989:2022)

Confidence Calibration: The degree to which reported confidence matches actual prediction accuracy; a calibrated system reporting 90% confidence is correct approximately 90% of the time.

Correlated Failure: Failure of multiple models on the same inputs due to shared training data or architecture, undermining the independence assumption of ensembles. (Framework-defined)

Quantization: Reduction of the numerical precision of model parameters for deployment; effects vary by layer and require revalidation.

Knowledge Distillation: Training a smaller student model to reproduce a larger teacher model's behavior; the student's behavior on edge cases must be independently validated.

ODD Evolvability: Whether, and within what bounds, a system's operational design domain is permitted to change through continuous learning. (Framework-defined)

System Classification Terms

Safety-Critical AI: AI outputs directly affect human safety or system survivability. (Framework Section 3.3)

Mission-Critical AI: AI outputs affect operational success but not human safety. (Framework Section 3.3)

Operational Support AI: AI supports operations but does not drive critical decisions. (Framework Section 3.3)

Appendix B: AI Classification Checklist

B.1 AI Presence Determination

1. Does the system use behavior derived from training data rather than explicitly programmed logic?

2. Are system outputs probabilistic or non-deterministic?

3. Can system behavior change or degrade over time without code modifications?

4. Does the system make or inform decisions affecting human safety or operational success?

Evaluation: If any answer is Yes, AI is present; proceed to B.2. If all answers are No, standard software assurance applies.

B.2 Safety-Critical AI Determination

5. Do AI outputs directly affect human safety or system survivability?

6. Could AI malfunction result in loss of life or critical equipment?

7. Is the AI in the control path for safety-critical functions?

Evaluation: If any of 5-7 is Yes, classify as Safety-Critical AI. If all are No, proceed to B.3.

B.3 Mission-Critical AI Determination

8. Do AI outputs affect operational success (but not human safety)?

9. Could AI malfunction result in loss of operational objectives?

10. Is the AI in the decision path for mission-critical functions?

Evaluation: If any of 8-10 is Yes, classify as Mission-Critical AI. If all are No, classify as Operational Support AI.

B.4 Requirement Applicability

Classification	Applicable Requirements
Safety-Critical AI	All ten core areas (AI-1 through AI-10); AI-11, AI-12, AI-13 wherever applicable; no tailoring without PSRB approval
Mission-Critical AI	AI-1 through AI-6, AI-8, AI-9, AI-10, with AI-7 tailored (CE approval); AI-11, AI-12, AI-13 as applicable
Operational Support AI	AI-1 through AI-6 and AI-10 minimum; AI-7, AI-8, AI-9 as tailored (SA authority); AI-11, AI-12, AI-13 as applicable

Table 6. Requirement applicability by classification.

B.5 Conditional Requirement Determination

1. Does the system process personal data under any applicable privacy framework (GDPR, HIPAA, CCPA/CPRA, or analogues)? If Yes, AI-10 applies in full; if No, document AI-10 sub-requirements as Not Applicable with rationale.
2. Does the system incorporate two or more interacting AI models (cascaded, fused, ensemble, or retrieval-augmented)? If Yes, AI-11 applies.

3. Does the system incorporate neural network components? If Yes, AI-12 applies.
4. Does the system update model parameters during operation? If Yes, AI-13 applies.

Evaluation: Conditional requirement areas attach to the classification determined in B.2 and B.3; they do not change it, with one exception: a system that learns during operation meets the potential-drift criterion and is classified no lower than Mission-Critical.

Appendix C: References

RTCA. (2011). DO-178C: Software Considerations in Airborne Systems and Equipment Certification.

ISO. (2018). ISO 26262: Road Vehicles - Functional Safety.

NASA. (2019). NPR 7150.2D: NASA Software Engineering Requirements.

NASA. (2022). NASA-STD-8739.8B: Software Assurance and Safety Standard.

NASA. (2023). NPR 8000.4C: Agency Risk Management Procedural Requirements.

ISO/IEC. (2022). ISO/IEC 22989:2022: Information Technology - AI Concepts and Terminology.

ISO/IEC. (2023). ISO/IEC 5338:2023: Information Technology - AI System Life Cycle Processes.

ISO/IEC. (2022). ISO/IEC 23053:2022: Framework for AI Systems Using Machine Learning.

ISO/IEC. (2021). ISO/IEC TR 24027:2021: Bias in AI Systems and AI Aided Decision Making.

ISO/IEC. (2020). ISO/IEC TR 24028:2020: Overview of Trustworthiness in Artificial Intelligence.

ISO/IEC. (2021). ISO/IEC TR 24029-1:2021: Assessment of the Robustness of Neural Networks.

ISO/IEC. (2023). ISO/IEC 23894:2023: Artificial Intelligence - Guidance on Risk Management.

ISO/IEC. (2023). ISO/IEC 42001:2023: Artificial Intelligence - Management System.

ISO/IEC. (2023). ISO/IEC 25059:2023: Quality Model for AI Systems.

ISO/IEC. (2025). ISO/IEC TS 6254: Objectives and Approaches for Explainability and Interpretability of ML Models and AI Systems.

ISO/IEC. (2019). ISO/IEC 27701:2019: Privacy Information Management.

ISO/IEC. (2011). ISO/IEC 29100:2011: Privacy Framework.

ISO. (2022). ISO 21448:2022: Road Vehicles - Safety of the Intended Functionality.

ISO/SAE. (2021). ISO/SAE 21434:2021: Road Vehicles - Cybersecurity Engineering.

ISO. (2023). ISO 34503: Road Vehicles - Taxonomy for Operational Design Domain.

SAE. (2021). J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems.

UNECE. (2021). Regulation No. 157: Automated Lane Keeping Systems.

EUROCAE/SAE. (In development). ED-324 / ARP6983: Process Standard for Development and Certification of Aeronautical Safety-Related Products Implementing AI.

NIST. (2023). NIST AI 100-1: AI Risk Management Framework.

NIST. (2023). NIST AI 100-2e2023: Adversarial Machine Learning - A Taxonomy and Terminology of Attacks and Mitigations.

NIST. (2024). NIST AI 600-1: AI Risk Management Framework: Generative AI Profile.

NIST. (2022). NIST SP 1270: Towards a Standard for Identifying and Managing Bias in Artificial Intelligence.

NIST. (2020). NIST Privacy Framework v1.0.

NIST. (2010). NIST SP 800-122: Guide to Protecting the Confidentiality of Personally Identifiable Information.

European Union. (2024). Regulation (EU) 2024/1689: Artificial Intelligence Act.

European Union. (2016). Regulation (EU) 2016/679: General Data Protection Regulation.

U.S. Department of Health and Human Services. HIPAA Privacy and Security Rules, 45 CFR Part 164.

State of California. CCPA/CPRA, Cal. Civ. Code §§ 1798.100 et seq.

FDA. (2023). Marketing Submission Recommendations for a Predetermined Change Control Plan for AI/ML-Enabled Device Software Functions.

FAA. (2024). Roadmap for Artificial Intelligence Safety Assurance.

IEEE. (2016). IEEE 1012-2016: Standard for System, Software, and Hardware Verification and Validation.

Williams, K. (2026). AI Requirements Framework for Safety-Critical Systems, Version 3.0. Safety Critical Labs. DOI: 10.5281/zenodo.19501092.

Appendix D: Source Authority and Traceability

Terminological Authority

Every term used in this framework traces to an authoritative source. Terminology is verified against ISO/IEC standards and NIST publications with specific section references (ISO/IEC 22989:2022 for core AI concepts, ISO/IEC 5338:2023 for lifecycle terms, NIST AI 600-1 for generative AI failure modes), with additional terms drawn from recognized academic sources in machine learning and human factors. Terms with no authoritative upstream source, including the system classification tiers (Safety-Critical AI, Mission-Critical AI, Operational Support AI), are explicitly framework-defined.

This traceability is not incidental. Regulatory frameworks derive authority in part from the precision and provenance of their terminology. When this framework defines "hallucination" as "the production of confidently stated but erroneous or false content," that definition traces to NIST AI 600-1 Section 2.4. When it defines "explainability" as "the property of an AI system to express important factors influencing results in a way that humans can understand," that traces to ISO/IEC 22989:2022 Section 3.5.7. When it defines "continuous validation" as an ongoing monitoring process during operation, that traces to ISO/IEC 5338:2023 Section 6.4.14.

This deliberate alignment serves three purposes. It ensures the framework uses terms consistently with international standards, reducing ambiguity in interpretation. It provides regulatory traceability: an assessor can verify that the framework's requirements are grounded in recognized standards rather than invented terminology. And it enables integration with other standards that reference the same ISO/IEC definitions, supporting adoption across domains and regulatory jurisdictions.

Source Authority Typology

Version 3.0 classifies every piece of framework content against its upstream sources using a five-category typology, applied across all requirements and maintained in future revisions. Where a concept blends categories, the dominant relationship is named.

Direct adoption uses source content verbatim: the IEEE 1012-2016 verification methods (inspection, analysis, demonstration, test) that anchor the verification architecture.

Adaptation modifies a source for the AI/ML context: the operational design domain concept, adapted from SAE J3016's automotive definition into a domain-agnostic AI ODD declaration.

Synthesis combines sources where no single standard covers the territory: the consequence-based classification tiers draw on NPR 8000.4C consequence classification, ISO 26262 ASILs, and DO-178C design assurance levels; the continuous learning requirements synthesize ISO/IEC 22989's continuous-learning provisions, ISO/IEC 23894's risk management for evolving systems, and the framework's own ODD evolvability concept.

Extension adds new content where a source recognizes the territory but not the AI-specific case: extending traditional lifecycle operations phases to host AI-specific continuous activities.

Framework-defined marks novel content with no upstream standard, such as the distinction between the population a system acts upon and the population that operates it.

Verification Status of Cited References

Inline domain-standard citations throughout the framework have been verified at clause level against source documents to the extent those documents are available in the certification project library. Five commercially distributed standards are cited but were not available for clause-level verification at the time of the Version 3.0 release: RTCA DO-326A, DO-330, DO-355, and DO-356A, and AAMI HE75. Their citations are retained as informative cross-references because they represent the recognized authoritative standards in their respective domains, and they are flagged for re-verification when source documents become accessible. Because inline domain citations are informative rather than normative, their inclusion does not constitute a normative dependency of the framework.

A complete term-by-term traceability matrix, documenting each term's primary source, specific section reference, and verification status, is maintained as a companion reference in the full requirements document.