SAFE-V: A Systems Safety Framework for Artificial Intelligence Adapting ISO 26262 Functional Safety and SOTIF Methodology to a V-Model AI Lifecycle

Jherrod Thomas
Independent Researcher
Jherrod.thomas@aol.com | www.jherrodthomas.com

***Abstract*—Current artificial intelligence (AI) safety practice relies on ad-hoc red-teaming, post-hoc evaluation, and voluntary governance frameworks that provide neither the rigor nor the auditable evidence expected of other safety-critical domains. Decades of maturation in automotive, aerospace, medical, and industrial functional safety—codified by ISO 26262, IEC 61508, DO-178C, IEC 62304, and most recently by ISO/PAS 21448 (SOTIF) and UL 4600—offer a transferable engineering methodology that AI has not yet systematically adopted. This paper specifies SAFE-V (Safety Assurance Framework for AI via the V-model), a systems-engineering safety framework that adapts Hazard Analysis and Risk Assessment (HARA), Safety Integrity Levels, Functional and Technical Safety Concepts, Fault Tree and Failure Modes Effects Analysis, Systems-Theoretic Process Analysis, and Goal-Structuring-Notation assurance cases to the particular failure modes of AI. SAFE-V introduces an AI-specific integrity level (AISIL) parameterized by severity, exposure, controllability, and epistemic uncertainty; a dual left-arm V-model that runs classical safety engineering in parallel with an ML engineering arm for data, model, and operational-design-domain governance; and a right-arm verification and validation cascade that binds unit-level behavioral testing, integration-level runtime-assurance monitors, system-level simulation and red-teaming, and post-deployment field monitoring to the left-arm requirements through bidirectional traceability. We position SAFE-V against the NIST AI Risk Management Framework, ISO/IEC 42001, UL 4600, AMLAS, and the EU AI Act, and propose a certification-grade evaluation protocol built on ML-adapted FMEA, SOTIF-style triggering-condition coverage, and Leveson's STPA.**

Index Terms*—*AI assurance case, AI safety, functional safety, hazard analysis, ISO 26262, ISO 21448 SOTIF, machine learning assurance, runtime assurance, safety integrity level, STPA, systems engineering, V-model.

I. Introduction

ARTIFICIAL INTELLIGENCE systems built on deep neural networks and large language models now operate in clinical triage, driver-assistance, industrial control, financial underwriting, and national-security applications whose failure can produce physical injury, economic loss, or infringement of civil rights [1], [2]. The engineering community's response has been a rapid proliferation of *governance* artifacts—the NIST AI Risk Management Framework [3], ISO/IEC 42001 [4], the EU AI Act [5], and voluntary red-team exercises [6]—together with a smaller literature on *machine-learning assurance* such as AMLAS [7] and ML assurance cases [8]. These artifacts establish *what* a trustworthy AI system should look like; they do not prescribe the engineering lifecycle by which it is actually built.

In contrast, the automotive, avionics, process-industries, and medical-device communities have, over forty years, converged on a mature, auditable engineering methodology anchored in IEC 61508 [9], ISO 26262 [10], DO-178C [11], and IEC 62304 [12]. This methodology is organized around the V-model [13]: the left arm traces system concept through hazard analysis, safety goals, architecture, and implementation; the right arm binds every implementation artifact to a corresponding verification or validation step; and the base of the V carries unit-level evidence. ISO/PAS 21448 (SOTIF) [14] extends this lifecycle to cover *performance insufficiencies*—hazards that arise not from component failure but from the system's inability to handle certain operating conditions. UL 4600 [15] applies analogous thinking to fully autonomous products. These standards are the closest existing fit to the dominant failure modes of modern AI, which are precisely failures of *performance* in rare or out-of-distribution conditions rather than failures of discrete components.

This paper specifies SAFE-V—**S**afety **A**ssurance **F**ramework for AI via the **V**-model—an engineering lifecycle framework that adapts the ISO 26262 and SOTIF methodology to AI. SAFE-V does not replace the NIST AI RMF or ISO/IEC 42001; rather, it supplies the missing engineering spine that governance artifacts presume but do

not specify. Its contributions are: (1) a formal AI hazard-analysis method (AI-HARA) that extends Severity-Exposure-Controllability classification with an epistemic Uncertainty axis appropriate to learned components; (2) an AI Safety Integrity Level (AISIL) scheme that assigns process rigor to hazards rather than imposing a single regime on a whole system; (3) a dual-arm V-model that runs a classical safety arm in parallel with an ML-engineering arm covering data, model, and operational-design-domain (ODD) concerns; (4) a catalogue of AI-specific safety mechanisms—runtime monitors, out-of-distribution detectors, uncertainty quantifiers, redundancy patterns, and safety-governor shells—mapped to ISO 26262–style diagnostic-coverage targets; (5) a verification and validation cascade that composes unit behavioral testing [16], differential testing [17], SOTIF triggering-condition coverage, and STPA [18]; and (6) an assurance-case template in Goal Structuring Notation [19] that ties the entire evidence base together.

Section II surveys the relevant functional-safety and AI-assurance literature. Section III formalizes the AI hazard model and SAFE-V's design goals. Section IV specifies AI-HARA and the AISIL determination. Section V presents the dual-arm V-model. Section VI catalogues AI-specific safety mechanisms and verification methods. Section VII describes the assurance-case structure. Section VIII works a case study in ML-based medical triage. Section IX discusses an evaluation protocol. Section X addresses threats to validity. Section XI concludes.

II. Related Work

A. Functional Safety Standards

IEC 61508 [9] defines safety lifecycle and Safety Integrity Levels (SIL 1–4) for electrical/electronic/programmable-electronic systems. ISO 26262 [10] specializes this lifecycle for automotive E/E systems, introducing Automotive Safety Integrity Levels (ASIL QM, A, B, C, D) derived from Severity, Exposure, and Controllability parameters; hardware metrics (SPFM, LFM, PMHF); and a twelve-part architecture covering management, concept, system, hardware, software, production, and supporting processes. DO-178C [11] governs airborne software with five Design Assurance Levels and coverage requirements up to Modified Condition/Decision Coverage. IEC 62304 [12] governs medical-device software with three software-safety classes. These standards share a common skeleton: identify hazards, allocate integrity requirements to functions, verify by constructive evidence, and assemble a safety case.

B. SOTIF and Autonomy

ISO/PAS 21448 (SOTIF) [14] addresses the hazards of *intended functionality*—situations in which a correctly implemented system nonetheless causes harm because its perception, reasoning, or actuation is insufficient for a particular *triggering condition*. SOTIF reorganizes the operating space into four regions: known-safe, known-unsafe, unknown-safe, and unknown-unsafe; the engineering objective is to shrink the unknown-unsafe region through scenario identification, triggering-condition coverage, and field evidence. UL 4600 [15] defines a goal-based safety case for fully autonomous products that integrates SOTIF-style arguments with classical hazard analysis. Koopman and Wagner [20] argue that self-driving safety must therefore combine SIL-style process assurance with operational-design-domain management and field feedback loops.

C. AI Governance Frameworks

The NIST AI RMF 1.0 [3] organizes trustworthiness around Govern, Map, Measure, and Manage functions and names validity, reliability, safety, security, resilience, accountability, transparency, explainability, privacy, and fairness as core properties. The NIST Generative AI Profile [21] adds 12 generative-specific risks. ISO/IEC 42001 [4] certifies AI management systems analogous to ISO 9001 quality systems. The EU AI Act [5] imposes tiered obligations on high-risk and general-purpose AI. These frameworks are prescriptive about outcomes but deliberately neutral about implementation methodology—they do not specify a V-model, nor a HARA, nor a safety case structure.

D. Machine-Learning Assurance

Kelly and Weaver [19] introduced Goal Structuring Notation (GSN) as a graphical language for safety cases; Hawkins [22] and the AMLAS methodology [7] adapt GSN to ML components, defining assurance patterns for data management, model learning, and model deployment. Ashmore, Calinescu, and Paterson [23] systematize ML assurance challenges into a four-stage lifecycle. Salay and Czarnecki [24] argue that ISO 26262 should be extended with ML-specific work products including ODD specification, data-quality requirements, and model-performance requirements. Picardi [25] proposes assurance-case patterns for ML. Burton [26] analyzes the mismatch between ISO 26262 software-unit assumptions and ML model behavior. SAFE-V draws on all of these and integrates them into a single prescriptive lifecycle.

E. ML Testing and Runtime Assurance

CheckList [16] imports behavioral testing from software engineering; DeepXplore [17] introduces neuron-coverage-guided differential testing; DeepTest [27] generates adversarial inputs for autonomous-driving DNNs. Zhang [28] surveys ML testing. Varshney [29] frames trustworthy ML around robustness, reliability, fairness, accountability, interpretability, and privacy. Runtime assurance patterns such as the Simplex architecture [30] and its AI-era adaptation—safety-governor shells that wrap a learned controller with a verified fallback—provide a template for guarding ML at deployment time. Out-of-distribution

detection [31] and uncertainty quantification supply the triggering signals for such shells.

F. Systems-Theoretic Safety

Leveson's STPA [18] reframes hazard analysis away from component-failure chains toward unsafe control actions in a control-theoretic model. STPA is particularly suited to AI because it treats software, learned components, operators, and organizations uniformly as controllers subject to inadequate control actions, process-model errors, and feedback failures. STPA-based analyses of autonomous systems [32] have identified hazards that component-centric FMEA missed, and SAFE-V adopts STPA as a mandatory complement to FMEA for any function instantiated by a learned component.

III. Problem Formulation

A. AI Hazard Model

We classify AI hazards by their causal origin. *Specification hazards* arise when the optimization objective does not capture the true goal (reward hacking, Goodhart's Law, sycophancy). *Distributional hazards* arise when operating inputs drift outside the training distribution. *Robustness hazards* arise under adversarial perturbation, natural corruption, or sensor degradation [29]. *Emergent hazards* arise when capability or behavior scales non-linearly with size [2]. *Interaction hazards* arise from control-loop couplings with humans, actuators, or other AI, and are best elicited by STPA [18]. *Assurance-decay hazards* arise from post-deployment degradation: data drift, model staleness, silent label shift, and feedback-loop amplification [33]. A complete AI-HARA must address all six classes; a framework that addresses only specification and robustness will be blind to the hazards that dominate field incidents.

B. Why Classical Functional Safety Is Necessary but Insufficient

ISO 26262 assumes, implicitly, three properties that are violated by modern ML: (i) specifications are complete and testable; (ii) component failures are discrete and independent; (iii) verification at unit level composes to system-level confidence. Learned components violate all three: their specification is the training distribution; their failures are continuous and correlated; and unit-level accuracy does not compose into system safety under distribution shift. SOTIF [14] addresses (i) and partially (iii) but was written with rule-based perception in mind and lacks prescriptive work products for the ML engineering lifecycle. UL 4600 [15] addresses (iii) at system level but is goal-based rather than prescriptive. The gap, which SAFE-V fills, is a prescriptive lifecycle that combines ISO 26262's discipline with SOTIF's performance-insufficiency model and adds explicit ML-engineering work products.

C. Design Goals

G1. *Lifecycle completeness.* Every phase from concept through decommissioning has explicit work products, entry and exit criteria, and verification steps.

G2. *Bidirectional traceability.* Every hazard traces down to implementation and test evidence, and every artifact traces up to at least one hazard or safety requirement.

G3. *Rigor proportional to risk.* Process rigor, coverage, and independence scale with AISIL rather than being uniform across a system.

G4. *ML-native work products.* Data requirements, ODD specification, model-performance requirements, and runtime-monitor requirements are first-class citizens rather than retrofits.

G5. *Performance-insufficiency awareness.* SOTIF triggering-condition analysis is mandatory for any learned component.

G6. *Continuous assurance.* Safety cases are maintained as living artifacts with field-monitoring feedback loops rather than as one-time certification snapshots.

G7. *Standard-agnostic.* The framework maps cleanly to ISO 26262, IEC 61508, DO-178C, IEC 62304, and IEC 61511 so that domain-specific certification can adopt SAFE-V without abandoning an existing regulatory regime.

IV. AI-HARA and AISIL

A. AI-HARA Procedure

AI-HARA adapts the ISO 26262 HARA procedure [10] by augmenting the Severity-Exposure-Controllability classification with an Uncertainty axis and by replacing the automotive-specific operational situations with an explicit Operational Design Domain (ODD) specification [20]. The analyst (i) defines the item and its ODD; (ii) enumerates item functions and the hazardous events that each function can cause; (iii) classifies S, E, C, U; (iv) derives an AISIL; (v) defines a safety goal and a corresponding Fault Tolerant Time Interval (FTTI); and (vi) records the result in a HARA ledger whose entries are unique, traceable, and maintained across model updates.

B. Severity, Exposure, Controllability

Severity S0–S3 follows ISO 26262: no injury, light/moderate injury, severe/life-threatening injury, fatal injury. For non-safety domains (privacy, fairness, financial), the severity rubric is re-parameterized but the four-level structure is preserved. Exposure E0–E4 is the fraction of operating time in which the triggering situation occurs. Controllability C0–C3 is the ability of a human supervisor, or an engineered safety mechanism, to avert harm once the hazard manifests.

C. Epistemic Uncertainty Axis

ML-specific hazards are modulated by how well the operating input is covered by training evidence. Following Hora and Kocaoglu [34] and the SOTIF notion of unknown-unsafe regions, we define U0–U3: U0 the hazard occurs only on inputs closely matched to training data; U1 similar distribution with mild covariate shift; U2 significant distribution shift or known-rare triggering condition; U3 input is out-of-distribution or adversarially crafted. U is estimated from the retrieval overlap between an operating input and the training-data manifest, the model's epistemic-uncertainty estimate [35], and an out-of-distribution detector [31]. The U axis is the formal SAFE-V mechanism for importing SOTIF's unknown-unsafe region into a classical integrity-level calculation.

D. AISIL Determination

AISIL is derived from the four-tuple (S, E, C, U) through the matrix of Table I. An entry QM (Quality Managed) denotes hazards for which ordinary ISO 9001–style quality practice is sufficient; AISIL-A through AISIL-D denote progressively greater rigor, analogous to ASIL-A through ASIL-D but calibrated against learned-component failure rates rather than random hardware failure. AISIL-D thus requires independent verification, formal or semi-formal methods where applicable, a conservative fallback controller, and a safety case that is maintained continuously.

E. Safety Goals and FTTI

Each hazardous event yields at least one safety goal of the form "the system shall not [unsafe effect] under [ODD conditions] within [FTTI]." FTTI is inherited from ISO 26262 [10] and decomposes into detection time, reaction time, and safe-state-transition time. For AI systems with human oversight, the FTTI budget must account for human-in-the-loop latency; for fully autonomous systems, it must be covered by engineered safety mechanisms.

V. The SAFE-V V-Model

A. Dual-Arm Structure

The classical V-model [13] is preserved on the *safety arm*: concept → HARA → Functional Safety Concept → Technical Safety Concept → architecture → design → implementation → unit test → integration test → system validation → safety case. In parallel, SAFE-V introduces an *ML engineering arm* that carries out: ODD specification → data requirements → data collection and labeling → data validation → model architecture and training → model evaluation on held-out and OOD data → model integration → operational-model monitoring. The two arms are interleaved by a fixed set of mandatory cross-links (Fig. 1): hazards inform data-coverage requirements; ODD coverage gaps feed back into HARA; model-performance specifications become Technical Safety Requirements; runtime monitors instantiate the safety mechanisms called out in the Technical Safety Concept.

B. Left Arm: Decomposition

The left arm produces five refinement layers. *Item definition* specifies boundary, interfaces, and environment. *HARA* produces hazards, safety goals, and AISIL. *Functional Safety Concept (FSC)* allocates safety goals to architectural functions and defines the degradation ladder. *Technical Safety Concept (TSC)* refines the FSC into detectable fault conditions, diagnostic coverage targets, and safe-state definitions; for AI systems the TSC must include (i) a model-performance specification per function, (ii) an ODD envelope, (iii) an OOD-detection budget, and (iv) a fallback policy. *Detailed Design* specifies the model family, training protocol, hyperparameters, guardrails, and runtime monitor parameters.

C. ML Engineering Arm

The ML arm produces complementary work products. *Data Safety Requirements Specification (DSRS)* defines representativeness, balance, label quality, freshness, and adversarial-robustness requirements, derived from HARA outputs per AMLAS [7]. *Data provenance and versioning* use content-addressable storage with cryptographic commitment so that the exact training set supporting any production model is auditable [33]. *Model performance specification* expresses requirements as per-slice metrics (worst-group accuracy, expected calibration error, fairness disparities) rather than single aggregate numbers; acceptance bounds are AISIL-dependent. *Training-pipeline verification* includes reproducibility checks, hyperparameter change control, and deterministic training where feasible.

D. Right Arm: Verification and Validation

The right arm binds each left-arm artifact to an explicit verification or validation step. *Unit level*: behavioral test suites per CheckList [16] with capability-based coverage, fairness probes, and adversarial probes. *Integration level*: runtime-assurance monitors are injected into the system under test, fault-injection campaigns exercise the OOD-detection and fallback paths, and differential testing [17] compares the learned component to a reference oracle where one exists. *System level*: scenario-based simulation against a SOTIF triggering-condition catalogue, adversarial red-teaming [6], and end-to-end hardware/software-in-the-loop trials. *Acceptance level*: shadow-mode and champion-challenger deployments with predefined stopping rules, followed by staged roll-out with continuous monitoring.

E. Post-Deployment: The Extended V

Because AI systems degrade in ways that traditional electronic hardware does not, SAFE-V extends the V-model into a continuous-assurance loop. The right arm does not terminate at release but continues through field-monitoring

work products: data-drift monitors, label-shift detectors, concept-drift detectors, performance-regression alarms, and incident-response runbooks. Triggering events on these monitors flow back up the V and can force re-entry at any earlier phase—DSRS update, re-training, re-HARA, or hazard-log revision. This is analogous to the ISO 26262 Part 7 production and field-monitoring phase but is materially more active for AI.

VI. Safety Mechanisms and Metrics

A. Mechanism Catalogue

SAFE-V maps ISO 26262 safety-mechanism categories—detection, reaction, and mitigation—to AI-native instantiations.

• *Input monitors*: OOD detectors [31]; adversarial-example detectors; sensor-plausibility checks; prompt-injection filters for LLM inputs.

• *Output monitors*: calibrated-uncertainty thresholds [35]; hallucination detectors and claim-entailment checkers; fairness-slice monitors; rule-based safety filters.

• *Redundancy*: heterogeneous-model ensembling; dissimilar-architecture diverse redundancy [26]; rule-based or classical-controller fallback.

• *Supervision*: safety-governor shells in the Simplex pattern [30]; human-in-the-loop escalation for high-U inputs; authority limits that cap model action space.

• *Reaction*: degraded-mode transitions; controlled shutdown; rollback to prior model version; automatic deferral to human.

• *Post-deployment*: drift monitors; shadow deployments; incident-response automation; safety-case refresh triggers.

B. Diagnostic Coverage for AI

ISO 26262 requires that safety mechanisms achieve a claimed Diagnostic Coverage (DC) of the faults they address. For AI, the analogous concept is the probability that a hazardous situation is detected before the FTTI elapses. SAFE-V defines an Operational Diagnostic Coverage (ODC) as the conditional probability that, given an input falls in a hazardous region (either OOD or within a known-triggering subspace), the ensemble of input, output, and redundancy monitors flags the input within FTTI. ODC is estimated empirically from a curated hazardous-input corpus and stress-tested through fault injection. AISIL-C systems target ODC ≥ 97%; AISIL-D systems target ODC ≥ 99% with at least two independent detection paths.

C. Verification Rigor by AISIL

SAFE-V prescribes verification techniques by AISIL. AISIL-A accepts review, simulation, and statistical testing. AISIL-B adds structural coverage, fault injection, and differential testing. AISIL-C adds semi-formal verification of safety mechanisms, STPA analysis, and independent review. AISIL-D adds, where feasible, formal verification of safety-shell properties, mandatory runtime assurance, and independent assessment consistent with ISO 26262 Part 2 confirmation measures. The rigor ladder is inherited from the ISO 26262 Part 6 software-verification tables [10].

Algorithm 1: SAFE-V Lifecycle (item, ODD, policy)

```
Input:  item definition I; ODD spec O; AISIL policy P.
Output: safety case SC; deployed system D; monitors M.

// ---- Left arm: safety + ML engineering
H  <- AI-HARA(I, O)              // hazards + AISIL
SG <- SafetyGoals(H)             // with FTTI
FSC <- FunctionalSafetyConcept(SG)
TSC <- TechnicalSafetyConcept(FSC)
DSRS <- DataSafetyReqs(H, O)     // ML arm
MPS  <- ModelPerfSpec(TSC, DSRS, P)
A   <- Architecture(TSC, MPS)
SM  <- SafetyMechanisms(TSC, A)
// ---- Implementation
D_data  <- CollectAndValidate(DSRS)
D_model <- TrainAndEvaluate(A, D_data, MPS)
D_sys   <- Integrate(D_model, SM)
// ---- Right arm: verification
v_unit  <- UnitTest(D_sys, CheckList, Adversarial)
v_int   <- IntegrationTest(D_sys, FaultInject)
v_sys   <- SystemTest(D_sys, SOTIF_catalog,
RedTeam)
v_stpa  <- STPA(D_sys, H)        // mandatory
if not Passes(v_*, MPS, ODC_target(P)) then
   return Refine(H, DSRS, A, SM)   // loop
// ---- Safety case + deployment
SC <- GSN_AssuranceCase(H, SG, FSC, TSC, v_*)
M  <- DeployMonitors(SM, driftDetectors)
D  <- StagedRollout(D_sys, M, stopRules)
while operating do               // continuous
   if M.trigger() then           // drift/incident
      SC <- RefreshCase(SC, M.evidence)
      Re-enter(H or DSRS or Train)
return (SC, D, M)
```

VII. Assurance Case Structure

The SAFE-V safety case is expressed in Goal Structuring Notation [19]. The top-level goal G0 asserts: *the item is acceptably safe to operate within ODD O for its intended use.* G0 is decomposed by strategy S0 ("argument over hazards from AI-HARA and over the completeness of the hazard analysis") into sub-goals G1–Gn, one per hazard, plus GH (completeness of HARA). Each hazard sub-goal is supported by (i) a safety-requirement sub-argument tying the hazard to its safety goal and FSC/TSC; (ii) a verification sub-argument citing unit, integration, system, and STPA evidence; (iii) a mechanism sub-argument showing that safety mechanisms meet their ODC target; and (iv) a residual-risk sub-argument bounding the probability and severity of uncaught hazards. A parallel stack GM argues the ML engineering arm: data-quality goals per DSRS, model-performance goals per MPS, ODD-coverage goals, and post-deployment-monitoring goals. AMLAS [7] patterns are adopted verbatim for the ML sub-arguments, with the SAFE-V additions being (a) explicit ODC targets bound to AISIL and (b) explicit triggering-event rules that re-open the case.

An important constraint is that the assurance case is *living*. Each field-monitoring trigger—a drift alarm, a severe incident, a distribution-shift detection, a model update—creates a ledger entry that either (i) supplies new evidence that sustains the existing case, (ii) requires additional verification activity to restore the case, or (iii) forces a hazard re-analysis. The case is therefore a time-stamped, versioned artifact rather than a one-time certification snapshot, satisfying design goal G6 and aligning with UL 4600 [15].

VIII. Case Study: Medical Triage Assistant

We illustrate SAFE-V with a sketch of an ML-based medical-triage assistant deployed in an emergency department to recommend acuity levels. *Item definition*: the assistant ingests structured vitals, chief complaint, and free-text triage notes; it outputs an Emergency Severity Index level and a confidence estimate; a nurse retains final authority. *ODD*: adult patients, English-language intake, vitals from the hospital's two monitor vendors, chief-complaint taxonomy version 3.

AI-HARA identifies hazards including H1 (under-triage of a time-critical condition under atypical presentation), H2 (over-triage causing resource starvation), H3 (disparate accuracy across demographic subgroups), H4 (silent failure on inputs in a new language), and H5 (model staleness after practice-pattern change). Classifying H1 at S3/E2/C2/U2 yields AISIL-C; H3 at S2/E3/C2/U1 yields AISIL-B. Safety goals follow: SG1 "the system shall not recommend an acuity level lower than a clinician-defined floor for features matching Sepsis-Alert criteria within 30 seconds," with an FTTI decomposed as 3 s detection + 10 s UI alert + 17 s mandatory nurse override.

The FSC allocates SG1 to three mechanisms: a sepsis-feature rule-based monitor (independent of the ML model), an uncertainty-gated abstention behavior in the ML model, and a nurse-confirmation UI for any below-floor recommendation. The TSC fixes OOD-detection budget, ODC $\geq$ 97%, and a fallback to a rule-based acuity scorer on high-U inputs. The DSRS mandates balanced subgroup sampling, minimum counts per condition, and mandatory review for demographic-labeling quality. The MPS specifies per-subgroup sensitivity and specificity floors, expected calibration error ceilings, and a worst-group-accuracy bound. Verification includes CheckList behavioral suites for under-triage, DeepXplore-style differential tests against a clinical scoring rule, an STPA analysis of the nurse-model-UI control loop, and SOTIF-style scenario coverage including language and note-style shifts. Field monitoring tracks subgroup performance daily, triggers re-HARA on any pattern-of-care change, and rolls back to the last approved model on two consecutive weekly breaches of the MPS floors.

IX. Evaluation Protocol

A. Certification-Grade Metrics

A framework is credible only if its conformance can be measured. SAFE-V compliance is assessed on six axes: (1) HARA completeness, measured by the fraction of hazards surfaced by independent STPA that were also captured by AI-HARA; (2) AISIL distribution and its correlation with field-incident rate; (3) ODC estimated per safety mechanism from fault-injection campaigns; (4) SOTIF triggering-condition coverage, measured as the fraction of an independently curated triggering catalogue that is exercised in system testing; (5) safety-case health, measured by the fraction of top-level goals currently supported by unexpired evidence; (6) continuous-assurance responsiveness, measured by the median time from a monitor trigger to an evidence refresh or hazard-log update.

B. Benchmarks and Comparison

SAFE-V is evaluated comparatively against (i) a NIST AI RMF–only baseline; (ii) an ISO/IEC 42001 management-system baseline; (iii) an AMLAS-only baseline; and (iv) an UL 4600–only baseline. Evaluation items include an open-source perception-autonomy stack, an LLM-based clinical-support tool, and an industrial-control anomaly detector. Outcome metrics are hazard-coverage, time-to-detection of injected faults, subgroup-performance maintenance under drift, and certification-effort proxy. We hypothesize that SAFE-V matches or exceeds the union of the baselines while producing a single, coherent audit artifact.

C. Ablations

Planned ablations disable independently: the U axis in AISIL, the ML engineering arm, STPA, runtime-assurance monitors, and the continuous-assurance loop. Each is expected to degrade a specific metric—e.g., disabling U correlates with unknown-unsafe incidents; disabling runtime monitors degrades ODC; disabling continuous assurance permits silent drift—thus supporting the claim that the components are complementary.

X. Threats to Validity

Several threats merit explicit discussion. *Transfer validity.* ISO 26262 is automotive; applying its vocabulary to non-automotive domains requires domain-specific rubrics for S, E, C. SAFE-V supplies the skeleton; a domain must supply its own severity scale (clinical harm grades, privacy-harm categories, financial-loss tiers).

Measurement of U. The Uncertainty axis depends on reliable OOD detection and calibrated uncertainty, both of which remain open research problems [31], [35]. SAFE-V mitigates this by requiring multiple independent U estimators and by making U-dependent decisions conservative under disagreement.

Cost and organizational inertia. Full SAFE-V application is expensive. The AISIL mechanism is the principal cost

control: quality-managed (QM) hazards consume only ISO 9001–class effort.

Assurance-case gaming. Any goal-based argument is vulnerable to confirmation bias and to unfalsifiable rhetorical patterns; SAFE-V therefore requires explicit residual-risk sub-goals and mandatory STPA to surface control-loop hazards that argument-by-decomposition can miss.

Regulatory alignment. Domains already regulated (aviation, medical devices, automotive) have entrenched standards; SAFE-V must be framed as an augmentation rather than a replacement.

Model-update velocity. Modern ML pipelines retrain far more frequently than automotive software is released. Continuous-assurance tooling is therefore essential and is itself a non-trivial engineering investment.

XI. Conclusion

SAFE-V adapts the engineering methodology of functional safety—hazard analysis, integrity levels, V-model decomposition, systems-theoretic process analysis, runtime assurance, and goal-structured safety cases—to the distinctive failure modes of modern AI. The framework is prescriptive where existing AI governance is discretionary, ML-native where existing safety standards are silent, and explicitly integrated with SOTIF, UL 4600, AMLAS, the NIST AI RMF, and ISO/IEC 42001. The dual-arm V-model binds classical safety engineering to ML engineering through bidirectional traceability; AISIL scales rigor to risk; AI-HARA imports SOTIF's unknown-unsafe region into a classical integrity-level calculation via the epistemic-uncertainty axis; and the living assurance case sustains the safety argument across the continuous update cycles that characterize deployed AI. Future work includes (i) a reference toolchain that automates the ML-engineering arm and cross-links it to the classical arm; (ii) domain-specific S/E/C/U rubrics for healthcare, finance, and public-sector AI; (iii) formal semantics for the GSN assurance case so that machine-checked case refresh becomes tractable; and (iv) an open-source benchmark suite for SOTIF-style triggering-condition coverage of foundation-model deployments.

References

[1] D. Amodei *et al.*, "Concrete problems in AI safety," arXiv, Jul. 2016. [Online]. Available: https://arxiv.org/abs/1606.06565

[2] R. Bommasani *et al.*, "On the opportunities and risks of foundation models," Stanford CRFM, Stanford, CA, USA, Tech. Rep., Aug. 2021. [Online]. Available: https://arxiv.org/abs/2108.07258

[3] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST, Gaithersburg, MD, USA, NIST AI 100-1, Jan. 2023, doi: 10.6028/NIST.AI.100-1.

[4] International Organization for Standardization, *Information Technology—Artificial Intelligence—Management System*, ISO/IEC 42001:2023, Geneva, Switzerland, 2023.

[5] European Parliament and Council of the European Union, "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)," *Official Journal of the European Union*, L series, Jul. 2024.

[6] E. Perez *et al.*, "Red teaming language models with language models," in *Proc. Conf. Empir. Methods Nat. Lang. Process. (EMNLP)*, Abu Dhabi, UAE, Dec. 2022, pp. 3419–3448. [Online]. Available: https://arxiv.org/abs/2202.03286

[7] R. Hawkins, C. Paterson, C. Picardi, Y. Jia, R. Calinescu, and I. Habli, "Guidance on the assurance of machine learning in autonomous systems (AMLAS)," Univ. of York, York, U.K., Tech. Rep., 2021. [Online]. Available: https://arxiv.org/abs/2102.01564

[8] C. Picardi, C. Paterson, R. Hawkins, R. Calinescu, and I. Habli, "Assurance argument patterns and processes for machine learning in safety-related systems," in *Proc. Workshop Artif. Intell. Safety (SafeAI)*, New York, NY, USA, Feb. 2020, pp. 23–30.

[9] International Electrotechnical Commission, *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*, IEC 61508:2010, Geneva, Switzerland, 2010.

[10] International Organization for Standardization, *Road Vehicles—Functional Safety*, ISO 26262:2018, Geneva, Switzerland, 2018.

[11] RTCA, *Software Considerations in Airborne Systems and Equipment Certification*, DO-178C, Washington, DC, USA, Dec. 2011.

[12] International Electrotechnical Commission, *Medical Device Software—Software Life Cycle Processes*, IEC 62304:2006+A1:2015, Geneva, Switzerland, 2015.

[13] K. Forsberg and H. Mooz, "The relationship of system engineering to the project cycle," in *Proc. Nat. Council Syst. Eng. (INCOSE)*, Chattanooga, TN, USA, Oct. 1991, pp. 57–65.

[14] International Organization for Standardization, *Road Vehicles—Safety of the Intended Functionality*, ISO 21448:2022, Geneva, Switzerland, 2022.

[15] Underwriters Laboratories, *Standard for Safety for the Evaluation of Autonomous Products*, UL 4600, 2nd ed., Northbrook, IL, USA, 2022.

[16] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond accuracy: Behavioral testing of NLP models with CheckList," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Jul. 2020, pp. 4902–4912, doi: 10.18653/v1/2020.acl-main.442.

[17] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proc. 26th Symp. Oper. Syst. Principles (SOSP)*, Shanghai, China, Oct. 2017, pp. 1–18, doi: 10.1145/3132747.3132785.

[18] N. G. Leveson and J. P. Thomas, *STPA Handbook*. Cambridge, MA, USA: MIT, Mar. 2018. [Online]. Available: https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

[19] T. Kelly and R. Weaver, "The Goal Structuring Notation—a safety argument notation," in *Proc. Dependable Syst. Netw. Workshop Assurance Cases*, Florence, Italy, Jun. 2004.

[20] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE Int. J. Transp. Saf.*, vol. 4, no. 1, pp. 15–24, Apr. 2016, doi: 10.4271/2016-01-0128.

[21] National Institute of Standards and Technology, “Artificial Intelligence Risk Management Framework: Generative AI Profile,” NIST, Gaithersburg, MD, USA, NIST AI 600-1, Jul. 2024, doi: 10.6028/NIST.AI.600-1.

[22] R. Hawkins, I. Habli, T. Kelly, and J. McDermid, “Assurance cases and prescriptive software safety certification: A comparative study,” *Safety Science*, vol. 59, pp. 55–71, Nov. 2013, doi: 10.1016/j.ssci.2013.04.007.

[23] R. Ashmore, R. Calinescu, and C. Paterson, “Assuring the machine learning lifecycle: Desiderata, methods, and challenges,” *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1–39, Jun. 2021, doi: 10.1145/3453444.

[24] R. Salay and K. Czarnecki, “Using machine learning safely in automotive software: An assessment and adaption of software process requirements in ISO 26262,” arXiv, Aug. 2018. [Online]. Available: https://arxiv.org/abs/1808.01614

[25] C. Picardi, R. Hawkins, C. Paterson, and I. Habli, “A pattern for arguing the assurance of machine learning in medical diagnosis systems,” in *Proc. Int. Conf. Comput. Saf., Rel., Secur. (SAFECOMP)*, Lisbon, Portugal, Sep. 2019, pp. 165–179, doi: 10.1007/978-3-030-26601-1_12.

[26] S. Burton, L. Gauerhof, and C. Heinzemann, “Making the case for safety of machine learning in highly automated driving,” in *Proc. SAFECOMP Workshops*, Trento, Italy, Sep. 2017, pp. 5–16, doi: 10.1007/978-3-319-66284-8_1.

[27] Y. Tian, K. Pei, S. Jana, and B. Ray, “DeepTest: Automated testing of deep-neural-network-driven autonomous cars,” in *Proc. 40th Int. Conf. Softw. Eng. (ICSE)*, Gothenburg, Sweden, May 2018, pp. 303–314, doi: 10.1145/3180155.3180220.

[28] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, “Machine learning testing: Survey, landscapes and horizons,” *IEEE Trans. Softw. Eng.*, vol. 48, no. 1, pp. 1–36, Jan. 2022, doi: 10.1109/TSE.2019.2962027.

[29] K. R. Varshney, *Trustworthy Machine Learning*. Chappaqua, NY, USA: Independently Published, 2022. [Online]. Available: http://www.trustworthymachinelearning.com/

[30] L. Sha, “Using simplicity to control complexity,” *IEEE Softw.*, vol. 18, no. 4, pp. 20–28, Jul./Aug. 2001, doi: 10.1109/MS.2001.936213.

[31] J. Yang, K. Zhou, Y. Li, and Z. Liu, “Generalized out-of-distribution detection: A survey,” *Int. J. Comput. Vis.*, vol. 132, pp. 5635–5662, 2024, doi: 10.1007/s11263-024-02117-4.

[32] A. Abdulkhaleq, S. Wagner, D. Lammering, H. Boehmert, and P. Blueher, “A systematic approach based on STPA for developing a dependable architecture for fully automated driving vehicles,” *Procedia Eng.*, vol. 179, pp. 41–51, 2017, doi: 10.1016/j.proeng.2017.03.094.

[33] D. Sculley *et al.*, “Hidden technical debt in machine learning systems,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 28, Montreal, QC, Canada, Dec. 2015, pp. 2503–2511.

[34] E. Hüllermeier and W. Waegeman, “Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods,” *Mach. Learn.*, vol. 110, no. 3, pp. 457–506, Mar. 2021, doi: 10.1007/s10994-021-05946-3.

[35] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, Jun. 2016, pp. 1050–1059. [Online]. Available: https://arxiv.org/abs/1506.02142