ETIS WHITE PAPER SERIES

WP-003

Engineering Evidence

From Artifacts to Operational Proof in the AI Era

William T. O'Connell, Ph.D.

Adjunct Professor of Computer Science, Loyola University Chicago
Former IBM Distinguished Engineer and CTO, IBM Consulting Quality and Delivery
Creator, Engineering Trustworthy Intelligent Systems (ETIS)

July 2026

ETIS WP-003 | Version 1.0

Executive Abstract

AI can now generate code, tests, documentation, designs, plans, and operational artifacts at a speed that was recently unimaginable. Yet faster artifact production does not create trustworthy systems. It can instead increase the distance between what a team produces and what it can responsibly explain, validate, release, operate, and defend. In that environment, engineering evidence becomes a first-class product of the software life cycle.

Engineering evidence is the connected, reviewable record that supports a consequential engineering claim. It answers questions such as: What problem was authorized? What assumptions shaped the design? Why was this architecture selected? What changed? Who or what produced the change? How was it challenged? Which tests and evaluations support the release? What risks remain? How will the system be observed, contained, recovered, and improved? A document, test result, log, or approval becomes evidence only when it is relevant to a claim, traceable to an authoritative source, generated by a credible process, and preserved in context.

This paper argues that the AI era requires a shift from artifact-centered delivery to evidence-centered engineering. The shift is reinforced by NIST guidance on AI risk management, secure software development, documentation, provenance, testing, and continual monitoring; SLSA guidance on verifiable build provenance; DORA research on delivery outcomes and instability; OpenTelemetry's vendor-neutral model for operational signals; ISO/IEC 42001's management-system approach to accountable AI governance; and emerging agentic-security guidance. These sources address different concerns, but together they reveal the same direction: trust must be supported by visible, durable, and continuously updated evidence. [1-9]

ETIS operationalizes this direction by treating the repository as the engineering record, claims as reviewable propositions, and evidence as a life-cycle chain from intent through operations and stewardship. The objective is not to maximize documentation. It is to produce the minimum sufficient evidence needed for responsible decisions, while ensuring that high-consequence claims can survive critical review and operational reality.

Core thesis: a working system is not proof of a trustworthy system. Trustworthy engineering requires evidence that connects intent, decisions, implementation, verification, release, operation, and accountability.

1. The Evidence Gap

Software organizations have always produced artifacts: requirements, plans, diagrams, source code, test reports, approvals, release notes, and operational logs. The persistent problem is not the absence of artifacts. It is the absence of a reliable chain of meaning among them. A requirement may exist without a test. A test may pass without covering the risk that matters. A pull request may be approved without explaining the architectural tradeoff. A release may be deployed without a reproducible record of what was built. A postmortem may identify a control failure without changing the engineering workflow that allowed it.

AI expands this evidence gap. Generative tools can create plausible artifacts faster than teams can determine whether those artifacts are correct, coherent, current, or relevant. The danger is not merely hallucinated text or defective code. It is false confidence created by volume and polish. A comprehensive-looking test plan can omit the most consequential failure mode. A sophisticated architecture diagram can document a design that was never implemented. An agent can modify multiple repositories, tests, configurations, and deployment assets while leaving reviewers unable to reconstruct the reasoning or authority behind the work.

DORA's recent research describes a verification tax: time saved in generation is often re-spent auditing, and a material share of developers report limited trust in AI-generated code. The implication is not that AI should be avoided. It is that accelerated creation must be matched by accelerated and better-structured verification. [10] The professional question shifts from "Did the team produce the artifact?" to "What claim does the artifact support, and why should a reviewer believe it?"

Engineering evidence closes that gap by preserving the relationship among claims, sources, methods, results, decisions, and outcomes. It makes review possible before release and reconstruction possible after failure. Evidence is therefore not paperwork added to engineering. It is the mechanism through which engineering becomes accountable.

2. From Artifacts to Claims

An artifact is an object. Evidence is an artifact interpreted in relation to a claim. Source code is an artifact; a reviewed change set, linked to an authorized issue and supported by passing tests, can be evidence that a requirement was implemented. A monitoring dashboard is an artifact; a defined service-level objective, measured over an agreed period with known gaps, can be evidence that a service is meeting an operational commitment. A model card is an artifact; validated evaluation results, data provenance, limitations, and deployment constraints can be evidence that a model is suitable for a bounded use.

This distinction prevents evidence-centered engineering from degenerating into document accumulation. The goal is not more artifacts. The goal is stronger claims. A claim should be explicit enough to challenge: "This release satisfies the approved requirements," "This change does not expand the agent's authority," "This model performs adequately for the intended population," "This service can recover within the required window," or "This residual risk has been consciously accepted by the accountable owner."

Each claim requires a proportionate evidence package. The package should identify the authoritative requirement or policy, the method used to evaluate the claim, the result, the limitations, the responsible reviewers, and the disposition. Weak evidence is not necessarily false; it is evidence whose relevance, provenance, completeness, or credibility is unclear. Strong evidence does not eliminate uncertainty. It makes uncertainty visible enough to govern.

Evidence is not the artifact itself. Evidence is the defensible relationship between an engineering claim and the authoritative facts, methods, results, and decisions that support it.

A practical evidence model

Claim	Authority	Method	Result	Disposition
What is asserted?	Which requirement, policy, risk owner, or decision authorizes the assertion?	How was it tested, measured, reviewed, or reproduced?	What happened, including exceptions and uncertainty?	Who accepted, rejected, deferred, or constrained the claim?
Release is ready	Approved scope and release criteria	Automated tests, security checks, review, rollback exercise	Results, failures, known limitations	Named release authority and conditions
Agent action is bounded	Tool and permission policy	Scenario evaluation, permission tests, audit	Allowed and denied actions, escalation	Approved authority envelope and

	review	behavior	monitoring

3. Evidence as a Life-Cycle Chain

Engineering evidence has the greatest value when it forms a continuous chain rather than a set of isolated deliverables. The chain begins with intent: the problem, expected outcome, constraints, stakeholders, and authority to act. It continues through requirements, architecture, planning, implementation, review, verification, release, operations, incident response, and stewardship. Each stage produces evidence and inherits obligations from the stages before it.

Traceability is the connective tissue. ISO/IEC/IEEE 29148 emphasizes requirements information and life-cycle processes; architecture standards emphasize documented concerns, viewpoints, and decisions; NIST AI RMF asks organizations to document intended purposes, risks, evaluations, limitations, monitoring, and accountability. [1, 11, 12] These bodies of guidance do not require one tool or repository layout, but they consistently assume that consequential decisions and outcomes can be connected.

In practice, the chain can be lightweight. An issue links to a requirement; a branch and pull request link to the issue; tests and evaluations link to the change; the release identifies the approved commit and generated artifacts; operational dashboards and incidents link back to the release; the postmortem creates new work and updates controls. The chain becomes stronger when links are enforced by workflow rather than reconstructed manually at the deadline.

The life-cycle model also prevents evidence from becoming stale. A requirement accepted in planning may be invalidated by operational learning. A model evaluation may no longer represent the production population. A security exception may expire. A runbook may not match the deployed architecture. Evidence-centered engineering treats evidence as living configuration: versioned, reviewed, and updated when the system or its context changes.

4. The Repository as the Evidence System

WP-002 established the repository as the engineering system of record. WP-003 extends that argument: the repository and its connected delivery environment should preserve the evidence chain, not merely the implementation. Issues capture authorized intent and uncertainty. Architecture decision records preserve tradeoffs. Pull requests present the engineering argument for a change. Automated workflows generate repeatable test, scan, and provenance evidence. Releases establish controlled baselines. Operational links, runbooks, incident records, and postmortems return production learning to the engineering record.

This model does not require every artifact to reside physically in Git. Enterprise requirements tools, service-management systems, model registries, observability platforms, policy engines, and audit systems may remain authoritative for their domains. Repository-centered evidence means that the engineering record contains durable references, identifiers, ownership, and enough context to navigate those systems without relying on memory or private conversation.

The distinction is particularly important for AI agents. An agent needs authoritative, machine-readable context: approved requirements, repository instructions, architecture boundaries, test commands, policy constraints, and evidence obligations. A reviewer needs provenance: what was requested, which context and tools were used, what changed, which checks ran, and what human judgment accepted the result. The same repository structure that improves human review becomes the control surface for agentic execution.

Evidence should be generated as a by-product of disciplined work. Requiring teams to recreate it immediately before a phase gate is both expensive and unreliable. The repository should make the compliant path the

easiest path: templates that ask for the right reasoning, workflows that link work automatically, checks that preserve results, and release processes that assemble the evidence package from authoritative sources.

Everything important should leave evidence, but not everything should become a document. The strongest evidence is created naturally by the engineering workflow and remains connected to the decision it supports.

5. AI and Agentic Engineering Raise the Burden of Proof

AI-assisted engineering changes both the producer of artifacts and the scale at which change can occur. A human engineer may use a model to draft code or tests. A coding agent may plan and execute a multi-step change. Multiple agents may modify application code, infrastructure, data pipelines, prompts, evaluations, and operational controls. Deployed agents may select tools and act within live environments. Each progression increases the authority delegated to software and therefore increases the evidence required to justify that delegation.

The evidence burden should be risk-proportionate. Low-risk suggestions may require ordinary review. Bounded multi-file changes require explicit task intent, source references, test evidence, and human acceptance. Changes to security controls, payment logic, safety mechanisms, permissions, or production infrastructure require stronger ownership and independent challenge. Runtime agents require documented authority boundaries, scenario evaluations, audit trails, monitoring, intervention points, and containment behavior.

NIST's AI RMF and Generative AI Profile emphasize documentation, human oversight, data provenance, testing, monitoring, and risk treatment across the AI life cycle. NIST's AI RMF Playbook asks organizations to document data sources and transformations, technical specifications, testing methods, metrics, performance outcomes, limitations, and information available to stakeholders. [1-3] OWASP's emerging agentic-security guidance focuses on agents that plan, use tools, retain memory, and take action. [9] These concerns move evidence beyond model accuracy. The system must demonstrate how authority is granted, constrained, observed, and revoked.

AI-generated evidence must also be treated carefully. A model can summarize logs, draft a risk analysis, or propose tests, but it cannot validate its own output simply by restating it. Evidence generated or interpreted by AI requires provenance, source access, reproducible methods, and accountable human review. AI may accelerate evidence production; it does not become the authority that accepts the engineering claim.

An authority-to-evidence progression

AI role	Typical authority	Minimum evidence	Additional controls as risk rises
Assist	Suggest or draft	Prompt/use disclosure, source review, ordinary verification	Sensitive-data restrictions; attribution
Coordinate	Organize tasks and propose workflow	Plan, authoritative context, task traceability, review record	Separation of duties; bounded tool access
Execute	Perform multi-step engineering change	Full provenance, changed assets, tests, policy checks, human acceptance	Independent review; protected areas; rollback
Operate	Act in a live system	Authority model, scenario evaluation, telemetry, audit,	Real-time policy enforcement; kill switch; incident evidence

intervention, recovery

6. Verification, Evaluation, and Behavioral Proof

Traditional software verification remains necessary: unit, integration, system, performance, security, and regression testing. Intelligent and agentic systems add a second dimension: behavioral evaluation under uncertainty. The system may generate different outputs, select different tools, or follow different paths across runs. Evidence must therefore address not only expected outputs but also decision quality, policy adherence, refusal behavior, escalation, authority boundaries, unsafe side effects, and recovery.

A single benchmark or demonstration is weak operational evidence. Evaluation should represent intended use, foreseeable misuse, edge conditions, affected populations, and the environment in which the system will operate. Results should include distributions and failure modes, not only averages. The evidence package should identify the model and configuration, prompts and policies, data and scenario versions, tool permissions, measurement methods, acceptance thresholds, and unresolved limitations.

NIST frames trustworthy AI through validity and reliability, safety, security and resilience, accountability and transparency, explainability, privacy, and fairness. [1] The GenAI Profile adds risks associated with confabulation, harmful content, information integrity, privacy, cybersecurity, and human-AI configuration. [2] Evidence-centered engineering translates those characteristics into explicit claims and repeatable evaluations. It also recognizes that not every characteristic carries equal weight in every system. The required proof follows the intended use and consequence.

Operational evidence completes the argument. Pre-release evaluation cannot reproduce every production condition. The release must therefore define what will be monitored, which indicators suggest degradation or harm, who is responsible for intervention, and how the system can be rolled back or constrained. Verification is not a gate that ends at deployment; it becomes a continuing process of measuring whether the system remains inside its accepted operating envelope.

A demo proves that a path worked once. Operational proof requires evidence that the system behaves acceptably across relevant conditions, that failures are visible, and that intervention is possible.

7. Provenance and the Integrity of Evidence

Evidence is only as credible as its provenance. Reviewers need to know where information came from, how it was transformed, which tools and environments produced it, and whether it has been altered. This is familiar in scientific research and regulated quality systems; it is becoming central to software delivery as build systems, dependencies, generated code, models, data, and automated agents complicate origin.

SLSA defines provenance as verifiable information describing where, when, and how a software artifact was produced. It promotes attestations over inference and progressively stronger protection against tampering. [4, 5] NIST SSDF calls for protecting the integrity of software releases and preserving provenance. [6] Software bills of materials provide component inventories, but an inventory alone does not prove that an approved process produced the deployed artifact. Evidence integrity requires both content and chain of custody.

The same principle applies to AI systems. Data provenance should identify sources, transformations, labels, augmentations, constraints, and dependencies. Model provenance should identify the model, version, fine-

tuning or adaptation, evaluation assets, configuration, and deployment context. Agentic provenance should preserve task intent, context sources, tool calls, permissions, intermediate actions, generated changes, and approvals. Without this record, organizations may know what they have but not why they should trust it.

Provenance should be generated automatically where possible and protected proportionate to consequence. Signed attestations, immutable logs, protected branches, controlled builders, identity-aware workflows, and retention policies improve evidence integrity. Human narratives remain necessary for judgment and tradeoffs, but they should reference machine-generated facts rather than substitute for them.

8. Operational Evidence and the Reality Test

Software earns or loses trust in operation. Reliability, recoverability, security, serviceability, cost, user impact, and unintended behavior cannot be fully established in design review. Operational evidence is therefore not a secondary reporting concern. It is the reality test for engineering claims.

OpenTelemetry defines a vendor-neutral approach to generating and collecting traces, metrics, and logs. [7] Google SRE practices connect service-level objectives, monitoring, incident response, capacity, release engineering, and learning. DORA measures delivery through throughput and instability, including change failure and deployment rework. [8, 13] Together these practices show that evidence must connect delivery decisions to production outcomes.

The most useful operational evidence is designed before release. Teams define the service or behavioral objectives, instrument the relevant paths, establish alert conditions, document runbooks and rollback, and assign ownership. The release record links to the deployed configuration. Incidents link back to the release and the engineering decisions that shaped it. Postmortems create corrective work and update controls. This closes the evidence loop.

Operational evidence must also be interpreted honestly. A green dashboard can hide missing telemetry. An average latency can conceal tail failures. An incident count can decline because reporting is discouraged. Evidence-centered organizations preserve the limitations of their measures, triangulate important claims, and avoid converting indicators into simplistic targets. The purpose is learning and responsible decision-making, not cosmetic assurance.

The final reviewer of every engineering claim is operational reality. Evidence-centered engineering prepares for that review and ensures that the result returns to the engineering record.

9. Governance, Assurance, and the Evidence Architecture

Governance depends on evidence, but many governance programs begin with controls and reports rather than claims and decisions. The result is often a compliance layer that collects artifacts without improving engineering judgment. Evidence-centered governance starts by identifying consequential decisions, accountable owners, required assurance, and the evidence needed to support them.

ISO/IEC 42001 establishes an AI management-system framework based on policy, objectives, risk management, documented processes, monitoring, internal audit, corrective action, and continual improvement. [14] NIST AI RMF organizes risk work around Govern, Map, Measure, and Manage. [1] Neither source prescribes a GitHub workflow or ETIS repository. Their value is in defining outcomes and management responsibilities. ETIS connects those outcomes to the engineering evidence architecture.

Three layers are required. Design-time evidence establishes intended use, requirements, architecture, risk, data, authority, and ownership. Delivery-time evidence establishes what changed, how it was reviewed and tested, how artifacts were produced, and why the release was accepted. Runtime evidence establishes actual behavior, incidents, overrides, user impact, and corrective action. Governance fails when one layer is absent or disconnected.

Assurance should be independent enough to challenge the claim and close enough to engineering to understand the evidence. Not every change requires a board. Many obligations can be implemented through ownership, peer review, automated policy, and risk-based escalation. High-consequence systems may require independent verification, model-risk review, safety cases, regulatory evidence, or formal acceptance. The architecture should make the path and authority visible.

Evidence architecture by decision

Decision	Representative claim	Evidence package	Accountable disposition
Design approval	The proposed system is fit for its intended use	Requirements, alternatives, architecture, risk, impact, authority boundaries	Design authority / risk owner
Change approval	The change is safe enough to integrate	Linked intent, diff, review, tests, scans, AI provenance, exceptions	Code owner / engineering lead
Release approval	The specific baseline is ready to operate	Build provenance, evaluation, known limitations, rollback, runbook, monitoring	Release authority / product owner
Continued operation	The system remains within accepted bounds	SLOs, behavioral telemetry, incidents, overrides, drift, user impact	Service owner / governance authority

10. Failure Modes: When Evidence Becomes Theater

Evidence-centered engineering can fail in predictable ways. The first is artifact inflation: teams generate more documents, screenshots, logs, and checklists without connecting them to claims. The second is retrospective assembly: evidence is recreated immediately before review, often from memory. The third is automation opacity: pipelines emit green checks without preserving methods, inputs, thresholds, or exceptions. The fourth is approval substitution: a signature is treated as proof even when the approver lacked relevant evidence.

AI creates additional failure modes. Generated narratives can make weak evidence appear complete. Agents may produce tests that validate their own assumptions. Model-generated summaries can omit contradictory source data. Evaluation sets can be optimized until they cease to represent operational use. Audit logs can record activity without explaining authority or consequence. The remedy is not more prose. It is stronger provenance, independent challenge, representative evaluation, and explicit limitations.

Another failure mode is metric capture. When evidence measures become performance targets, teams optimize the visible indicator: more tests rather than better risk coverage, more approvals rather than better review, fewer incidents rather than better reporting. Evidence architecture must preserve qualitative judgment and encourage dissent. A review that discovers a serious limitation is a successful review, not a delivery failure.

Finally, organizations may over-centralize evidence. A single enterprise evidence platform can improve consistency, but it can also become detached from the systems and teams that generate the facts. ETIS favors

connected evidence close to engineering work, with enterprise aggregation where useful. The engineering record remains navigable, while governance can view portfolio-level posture without replacing local authority and context.

11. A Maturity Model for Evidence-Centered Engineering

Organizations rarely move directly from fragmented artifacts to a complete evidence architecture. Maturity develops as teams improve the visibility, connection, automation, and decision value of evidence. The progression is not a certification scale; it is a practical way to identify the next constraint.

At the artifact stage, teams produce documents and test outputs, but evidence is difficult to locate and relationships are implicit. At the traceable stage, intent, changes, tests, and releases are linked. At the enforceable stage, workflow requires critical evidence and prevents unreviewed change. At the operational stage, release evidence connects to telemetry, incidents, and postmortems. At the adaptive stage, evidence quality and operational outcomes continuously improve requirements, policies, evaluations, and automation.

AI adoption should not outrun this maturity. The faster a system can generate and execute change, the more damaging fragmented evidence becomes. Organizations can introduce AI at every stage, but the delegated authority should remain within the evidence capacity of the engineering environment. This is an important management principle: autonomy should rise only as verification, provenance, observability, and intervention capability rise.

Evidence maturity progression

Level	Characteristics	Primary risk	Next move
1. Artifact	Deliverables exist; evidence is manual and scattered	Review depends on memory and heroics	Name claims and authoritative sources
2. Traceable	Intent, change, tests, release, and ownership are linked	Links exist but obligations are optional	Embed evidence in workflow
3. Enforceable	Required reviews, checks, ownership, and provenance	Controls may become ceremonial	Measure decision quality and exceptions
4. Operational	Release evidence links to telemetry, incidents, and learning	Production data may be incomplete or misleading	Improve instrumentation and feedback
5. Adaptive	Evidence continuously improves policy, evaluation, and design	Optimization can target metrics rather than outcomes	Preserve judgment, challenge, and stewardship

12. The ETIS Position

ETIS treats engineering evidence as a unifying discipline across software engineering, AI governance, security, operations, and stewardship. Its doctrine is simple: everything important leaves evidence; AI proposes and engineers verify; governance is architecture; context is control; the model is not the system; and a demo is not operational proof. These principles are not slogans layered over the life cycle. They define how the life cycle is implemented.

In ETIS, evidence begins with project intent and continues through requirements, planning, architecture, implementation, AI use, verification, release readiness, observability, security, incident response, postmortems, governance, and stewardship. The repository preserves the engineering record. Engineering

Stages ES-100 through ES-114 provide a progression. Templates and examples scaffold the work. Review gates challenge claims. The professional repository starter kit demonstrates how evidence can remain connected across the full system life cycle.

ETIS does not claim that every system needs the same volume of evidence. Evidence must be proportionate to consequence, uncertainty, novelty, reversibility, and delegated authority. A student project, internal workflow, medical decision-support system, and autonomous industrial agent do not require identical assurance. They do require the same intellectual discipline: identify the claim, authority, method, result, limitation, owner, and disposition.

This is the distinction between evidence-centered engineering and compliance accumulation. ETIS seeks evidence that improves decisions and survives review. It rejects both extremes: undocumented speed and bureaucratic volume. The target is a system whose engineering record is sufficiently complete that another qualified reviewer can understand what was built, why it was built that way, how it was validated, what risks remain, and how it will be operated responsibly.

ETIS position: trustworthy systems are not declared trustworthy. They are engineered so that consequential claims can be examined, challenged, reproduced, monitored, and revised.

13. Implications for Leaders, Engineers, and Educators

For executives, evidence-centered engineering changes the meaning of assurance. Dashboards and certifications remain useful, but they should be backed by navigable engineering evidence. Leaders should ask whether strategic claims can be traced to operational facts, whether AI authority is visible, whether release decisions are reproducible, and whether incidents change the system of work. Investment should prioritize evidence-generating platforms and practices, not reporting layers detached from delivery.

For engineering leaders, the priority is to design the evidence path. Define authoritative sources, ownership, workflow, minimum evidence, exception handling, retention, and operational feedback. Remove duplicate reporting. Automate factual evidence. Preserve human reasoning where judgment matters. Evaluate AI tools not only by generation speed but by the provenance, reviewability, integration, and controls they provide.

For engineers, evidence becomes part of professional craft. A pull request should explain intent and risk. A test should support a claim. An architecture decision should preserve alternatives and consequences. A release should identify what is known and unknown. An incident should produce learning that changes code, configuration, tests, runbooks, or policy. The responsibility is not to create more artifacts; it is to leave the system more understandable and governable than it was before.

For educators, AI makes this discipline urgent. Students can generate polished code and documentation before they understand the system. Courses should therefore assess reasoning, traceability, review, testing depth, AI disclosure, release readiness, and operational evidence. Students should experience the repository as an engineering record and learn that professional credibility comes from claims that can withstand review, not from artifact volume.

Conclusion: Evidence Is How Engineering Earns Trust

The software profession is entering an era in which the ability to produce artifacts is abundant. The scarce capabilities are disciplined intent, coherent architecture, critical review, credible verification, controlled release, operational learning, and accountable judgment. Engineering evidence connects those capabilities.

Evidence does not guarantee that a system is correct or safe. It makes the basis for trust visible. It allows reviewers to challenge assumptions before release, operators to understand the accepted baseline, incident responders to reconstruct what happened, governance bodies to make informed decisions, and future engineers or agents to inherit authoritative context. It converts trust from a marketing assertion into an engineering proposition.

The direction of standards and professional practice is clear: document risk and intended use, preserve provenance, integrate secure development, evaluate behavior, monitor operation, and improve continuously. ETIS adds the connective tissue: a repository-centered, evidence-centered life cycle in which every consequential decision leaves a reviewable record.

In the AI era, this discipline becomes more important, not less. AI can accelerate the creation of software and the execution of engineering work. It cannot absorb human accountability. The organizations that succeed will not be those that generate the most artifacts. They will be those that can explain, verify, operate, and improve what they build - and produce the evidence to prove it.

References

[1] National Institute of Standards and Technology. “Artificial Intelligence Risk Management Framework (AI RMF 1.0).” NIST, 2023. https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf

[2] National Institute of Standards and Technology. “Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile.” NIST, 2024. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

[3] National Institute of Standards and Technology. “AI RMF Playbook.” NIST AI Resource Center, 2024. https://airc.nist.gov/docs/AI_RMF_Playbook.pdf

[4] SLSA Community. “SLSA: Supply-chain Levels for Software Artifacts.” Linux Foundation, 2026. https://slsa.dev/

[5] SLSA Community. “SLSA Provenance.” Linux Foundation, 2026. https://slsa.dev/spec/v1.2/

[6] National Institute of Standards and Technology. “Secure Software Development Framework (SSDF), SP 800-218.” NIST, 2022. https://csrc.nist.gov/pubs/sp/800/218/final

[7] OpenTelemetry Authors. “OpenTelemetry Documentation.” Cloud Native Computing Foundation, 2026. https://opentelemetry.io/docs/

[8] DORA. “Software Delivery Performance Metrics.” Google Cloud, 2026. https://dora.dev/guides/dora-metrics/

[9] OWASP GenAI Security Project. “OWASP Top 10 for Agentic Applications.” OWASP, 2026. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/

[10] DORA. “Balancing AI Tensions: Moving from AI Adoption to Effective Use.” Google Cloud, 2026. https://dora.dev/insights/balancing-ai-tensions/

[11] ISO/IEC/IEEE. “Systems and Software Engineering - Life Cycle Processes - Requirements Engineering, 29148:2018.” ISO/IEEE, 2018. https://www.iso.org/standard/72089.html

[12] ISO/IEC/IEEE. “Systems and Software Engineering - Architecture Description, 42010:2022.” ISO/IEEE, 2022. https://www.iso.org/standard/74393.html

[13] Google. “Site Reliability Engineering.” Google, 2016. https://sre.google/sre-book/table-of-contents/

[14] International Organization for Standardization. “ISO/IEC 42001:2023 Artificial Intelligence Management System.” ISO, 2023. https://www.iso.org/standard/81230.html

[15] Cybersecurity and Infrastructure Security Agency. “Software Bill of Materials.” CISA, 2025. https://www.cisa.gov/sbom

[16] GitHub. “About Artifact Attestations.” GitHub Documentation, 2026. https://docs.github.com/en/actions/security-for-github-actions/using-artifact-attestations

[17] Open Source Security Foundation. “OpenSSF Scorecard.” OpenSSF, 2026. https://scorecard.dev/

[18] National Institute of Standards and Technology. “Secure Software Development Practices for Generative AI and Dual-Use Foundation Models, SP 800-218A IPD.” NIST, 2024. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218A.ipd.pdf

[19] DORA. “State of AI-Assisted Software Development 2025.” Google Cloud, 2025. https://dora.dev/dora-report-2025/

[20] GitHub. “About Rulesets.” GitHub Documentation, 2026. https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/managing-rulesets/about-rulesets

[21] GitHub. “About Code Owners.” GitHub Documentation, 2026. https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-code-owners

[22] National Institute of Standards and Technology. “Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations, NIST AI 100-2e2025.” NIST, 2025. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2025.pdf

[23] DORA. “Value Stream Mapping for Software Delivery.” Google Cloud, 2024. https://dora.dev/guides/value-stream-management/

[24] SLSA Community. “Source Provenance Requirements.” Linux Foundation, 2026. https://slsa.dev/spec/v1.2/source-requirements

[25] O’Connell, William T.. “ETIS WP-001: Engineering Trustworthy Software in the AI Era.” ETIS White Paper Series, 2026. https://etisframework.org

[26] O’Connell, William T.. “ETIS WP-002: Repository-Centered Engineering.” ETIS White Paper Series, 2026. https://etisframework.org

About the Author

William T. O'Connell, Ph.D. is an Adjunct Professor of Computer Science at Loyola University Chicago and the creator of the Engineering Trustworthy Intelligent Systems (ETIS) framework. His professional career spans more than four decades of software engineering, architecture, delivery, quality, and technology leadership, including service as an IBM Distinguished Engineer and CTO for IBM Consulting Quality and Delivery. His current work focuses on repository-centered engineering, engineering evidence, trustworthy intelligent systems, agentic software delivery, and the preparation of students and practicing engineers for increasingly AI-assisted engineering environments.

This white paper is part of the ETIS White Paper Series, which examines the engineering practices required to build, govern, review, operate, and sustain trustworthy software and intelligent systems.

Suggested citation: O'Connell, W. T. (2026). Engineering Evidence: From Artifacts to Operational Proof in the AI Era. ETIS White Paper Series, WP-003, Version 1.0.