ETIS WHITE PAPER SERIES

WP-011

Engineering Trust

How Trust Emerges from Architecture, Evidence, Governance, and Operational Reality

Core Thesis

Trust is not a feature that can be added after a system works. It is an earned system property that emerges when intent is explicit, authority is bounded, decisions are reviewable, evidence is credible, operations are observable, and accountable people can intervene when reality departs from expectation.

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-011 | Version 1.0

Executive Abstract

Trust has become one of the most frequently invoked—and least precisely engineered—properties of modern software and intelligent systems. Product teams promise trustworthy AI, secure platforms, reliable services, transparent decisions, and responsible automation. Yet trust is often treated as a communications objective, an ethics statement, or an accumulation of controls. None of those is sufficient. People and institutions can rationally rely on a system only when its claims are supported by evidence, its behavior remains within understood boundaries, its failures are visible and recoverable, and responsibility for consequential decisions is unmistakable.

This paper develops a systems-engineering account of trust. It distinguishes trust from trustworthiness, confidence, compliance, and reputation; explains why trustworthy behavior is an emergent property of the complete sociotechnical system rather than the model or application alone; and presents an integrated trust architecture spanning intent, requirements, architecture, data, context, identity, implementation, supply chain, review, release, operations, governance, and stewardship. The paper also examines the harder problem introduced by agentic systems: when software can plan, use tools, retain memory, coordinate with other agents, and act in operational environments, trust must be grounded in bounded authority, strong identity, observable behavior, intervention capability, and progressive evidence.

The central argument is straightforward: organizations should stop asking whether a technology is trustworthy in the abstract and instead ask what specific reliance is justified, for whom, under what conditions, with what evidence, and with what means of recovery. Engineering Trustworthy Intelligent Systems (ETIS) treats trust as an earned claim produced through disciplined engineering and continuously tested in operation.

Keywords—trustworthy systems, software engineering, intelligent systems, agentic AI, assurance, governance, evidence, operational readiness, resilience, accountability, observability, provenance.

Summary

Software is increasingly permitted to influence decisions, coordinate work, move money, alter records, control infrastructure, recommend treatment, manage identities, and act through tools. Artificial intelligence raises the stakes because a system may no longer execute only prewritten logic. It may infer intent, select among alternatives, assemble context, plan multiple steps, delegate work, and adapt its behavior to conditions that were not enumerated in advance. The resulting question is not merely whether the system performs well. It is whether people, organizations, and society are justified in relying on it.

Trust is commonly discussed as though it were a single quality. In practice, reliance is contextual. A user may trust a system to draft text but not to send it. An engineer may trust an agent to propose a patch but not to merge it. A bank may trust a model to prioritize an investigation but not to close an account. A hospital may trust an algorithm to surface a signal but not to make an irreversible clinical decision. Engineering trust therefore begins by specifying the reliance relationship: who or what is relying on the system, for which outcome, within which authority boundary, and under which operating conditions.

NIST describes trustworthy AI through multiple characteristics—validity and reliability, safety, security and resilience, accountability and transparency, explainability and interpretability, privacy enhancement, and fairness with harmful bias managed [1]. That plurality is important. Trust cannot be reduced to model accuracy, security certification, uptime, or regulatory compliance. These are necessary contributors, but they can conflict and they can fail independently. A highly accurate model may be unsafe in a particular context. A secure service may be unreliable. A transparent system may still be unfair. A compliant process may produce an operationally fragile product.

The engineering implication is that trust must be composed. It emerges from the interaction of architecture, evidence, governance, human judgment, and operational behavior. NIST's systems-security engineering guidance makes a similar point for security, treating it as an emergent property of a system rather than a

component feature [2]. ETIS extends this systems perspective across the wider trust problem. Trustworthiness is not demonstrated by intent alone; it is built through requirements, design, controlled change, review, validation, release discipline, telemetry, incident learning, and stewardship.

For leaders, this reframes the investment agenda. The goal is not to purchase a 'trust layer' or create a review board that approves AI initiatives. The goal is to build an engineering operating model in which trust claims are explicit, evidence is generated continuously, authority is proportionate to consequence, exceptions are visible, and operational feedback can change decisions. For engineers, it means designing not only functional behavior but also boundaries, observability, provenance, intervention, rollback, and accountability. For educators, it means teaching students that a working demonstration is not operational proof.

The Executive Test

A system deserves trust only to the extent that a specific reliance claim can survive informed challenge: What is being trusted? By whom? For what consequence? Under what conditions? What evidence supports the claim? What happens when the claim is wrong?

1. Trust Is Not a Feature

Organizations often approach trust as a feature to be added near the end of delivery: an explanation screen, a privacy notice, a security review, a model card, a compliance assessment, or a human approval button. Each can contribute value, but none creates trustworthiness on its own. A system that is poorly scoped, weakly architected, trained on unsuitable data, granted excessive authority, inadequately tested, or operationally opaque remains untrustworthy even when its interface uses the language of responsibility.

Trust is also not the same as user confidence. Confidence is a psychological state; it can be deserved or undeserved. Polished interfaces, fluent language, anthropomorphic design, and brand reputation can increase confidence without improving the underlying system. This is particularly dangerous in AI systems because linguistic fluency is easily mistaken for knowledge, judgment, or certainty. Engineering should therefore aim not to maximize trust, but to calibrate it. Users should rely on a system to the degree warranted by its evidence and limitations.

Nor is trust equivalent to compliance. Compliance demonstrates conformance to selected obligations at a point in time. Trustworthiness requires that the system continue to behave acceptably as data, dependencies, models, users, threats, and operational conditions change. ISO/IEC 42001 addresses this dynamic by requiring an organizational management system for the responsible development and use of AI, including continual improvement rather than one-time approval [3]. Compliance can support trust, but a certificate cannot replace engineering judgment or operational evidence.

Reputation matters, but it is retrospective and coarse-grained. A reliable vendor can release a defective product; a previously safe model can become unsuitable after a context shift; a well-governed team can make a poor decision. Trustworthy engineering therefore resists institutional shortcuts. It asks for evidence at the level of the system and decision, while using organizational reputation as one input rather than the conclusion.

From trust to justified reliance

A more useful definition is justified reliance. Reliance is justified when the expected benefits and risks are understood well enough for a stakeholder to permit the system to perform a defined role, within bounded conditions, with evidence proportionate to consequence. This definition makes trust actionable. It requires a subject, an object, a purpose, a context, an authority boundary, and an assurance basis.

The reliance may be narrow. A team can trust a code agent to create a draft test because every change is reviewed and executed in a sandbox. The same team may not trust that agent to change production

infrastructure. Trust can also be conditional: an autonomous workflow may operate below a financial threshold, during defined hours, against approved counterparties, and with automatic escalation when anomalies appear. Conditions are not signs of weak trust; they are the architecture through which trust becomes rational.

2. Trustworthiness Is a System Property

Trustworthy behavior emerges from the complete sociotechnical system. The model contributes capability, but the system also includes requirements, policies, data, context, prompts, retrieval sources, identity, permissions, software components, user interfaces, workflow, human reviewers, build systems, deployment infrastructure, telemetry, incident procedures, and organizational incentives. A defect in any of these can invalidate the trust claim.

NIST SP 800-160 describes systems security engineering as a multidisciplinary approach to engineering trustworthy secure systems across the lifecycle [2]. The important idea is not limited to security: properties such as safety, reliability, resilience, privacy, accountability, and fairness arise from interactions among components and processes. They cannot be guaranteed by inspecting a single artifact. ETIS therefore treats the model as one governed component within a larger engineered system.

This perspective explains why strong model benchmarks do not establish operational trust. Benchmark performance may not represent the production population, user behavior, data quality, adversarial pressure, tool permissions, latency constraints, or failure consequences. A model may pass an evaluation while the surrounding retrieval pipeline introduces stale information, the interface obscures uncertainty, the workflow bypasses review, or the runtime grants a tool more authority than intended.

It also explains why conventional software can be untrustworthy despite deterministic code. Reliability failures, insecure defaults, opaque dependencies, missing rollback paths, poor incident response, and undocumented decisions all reduce justified reliance. Intelligent systems intensify these concerns; they do not create them.

Trust dimension	Engineering question	Representative evidence
Validity and reliability	Does the system perform its intended function under relevant conditions?	Requirements, test results, evaluation sets, error analysis, SLO history
Safety	Can foreseeable harm be prevented, contained, or recovered from?	Hazard analysis, authority limits, safeguards, simulations, incident drills
Security and resilience	Can the system resist, withstand, recover from, and adapt to attack or compromise?	Threat models, secure design evidence, provenance, red-team results, recovery tests
Accountability and transparency	Can consequential actions and decisions be attributed and challenged?	Decision records, identities, approvals, logs, disclosures, exception history
Explainability and interpretability	Can affected people and reviewers understand the basis and limitations of behavior?	Explanations, trace context, model/system documentation, user studies
Privacy and fairness	Are data and outcomes handled consistently with rights, obligations, and intended impact?	Data lineage, access controls, impact assessments, subgroup analysis, remediation records

3. The Architecture of Trust

Trust is not produced by one control; it is composed through an architecture. That architecture should make five things explicit: intended reliance, authority boundaries, evidence paths, intervention mechanisms, and

accountable ownership. These elements connect the promise made to stakeholders with the system's real behavior.

Intended reliance begins in requirements. Teams should state not only what the system will do, but what users and downstream systems may safely assume. Requirements for an intelligent assistant might distinguish suggestions from decisions, identify prohibited actions, define acceptable uncertainty, specify data handling, and establish when escalation is mandatory. Without these statements, trust is left to interface cues and user inference.

Authority boundaries translate intended reliance into enforceable design. Identity, permissions, rate limits, transaction thresholds, approved tools, data access, network boundaries, and separation of duties determine what the system can actually do. CISA's Secure by Design guidance emphasizes that manufacturers should take ownership of customer security outcomes and make secure defaults a core business requirement [4]. In the trust architecture, safe and secure defaults are not optional usability choices; they are constraints on consequence.

Evidence paths show how a reviewer can move from a claim to the underlying record. A release claim should connect to requirements, architecture decisions, implementation changes, reviews, tests, provenance, risk acceptance, and operational readiness. Repository-centered engineering is valuable because it preserves these relationships close to the change. Evidence-centered engineering goes further by judging whether the record is credible, relevant, sufficient, current, and independent enough for the decision.

Intervention mechanisms acknowledge that no complex system is perfectly predictable. Stop controls, human takeover, rollback, degraded modes, circuit breakers, revocation, quarantine, and incident escalation are part of the trust design. A system that cannot be safely interrupted requires a higher level of evidence before it can be relied upon.

Accountable ownership completes the architecture. Every consequential capability, risk, exception, release, and operational decision needs an identifiable owner with the authority and competence to act. Accountability cannot be assigned to 'the model,' 'the algorithm,' or a diffuse committee. It belongs to people and institutions.

Trust Architecture

Intent defines what reliance is acceptable. Architecture bounds what the system can do. Evidence supports what is claimed. Operations reveal what is true. Governance determines who may decide. Stewardship changes the system when the evidence changes.

4. Trust Claims Require Engineering Evidence

Trust is often weakened by vague claims: the system is secure, responsible, compliant, explainable, or production-ready. These labels compress many assumptions into language that is difficult to test. A professional trust claim should be specific enough to falsify. It should identify the behavior, population, operating conditions, consequence, time horizon, and accepted residual risk.

Evidence is the basis for accepting or rejecting that claim. NIST's AI RMF encourages organizations to govern, map, measure, and manage AI risks through a repeatable lifecycle [1]. The framework is intentionally flexible; engineering teams must translate its outcomes into concrete evidence. That evidence may include architecture analysis, test results, model evaluations, data lineage, human-factors studies, red-team findings, operational telemetry, incident records, and documented decisions.

Evidence quality matters more than evidence volume. Automatically generated test suites can contain many tests yet omit the important failure mode. A model card can be complete while describing an evaluation irrelevant to the deployment context. A dashboard can be green because it does not measure user harm. A

review can be documented even though the reviewer lacked independence or time. Trustworthy assurance examines the provenance, relevance, completeness, and challengeability of evidence.

Supply-chain provenance is an important example. SLSA defines provenance as verifiable information about where, when, and how a software artifact was produced [5]. This allows consumers and delivery systems to verify properties of the build rather than relying only on the artifact's appearance. The same logic should apply to models, datasets, prompts, retrieval corpora, tools, and agent configurations. When an independently changeable element can affect behavior, its origin and transformation history are part of the trust claim.

Evidence should also include negative knowledge: known limitations, unresolved findings, failed experiments, rejected alternatives, and conditions under which confidence falls. Hiding uncertainty may improve short-term acceptance but destroys calibrated trust. Radical transparency does not mean publishing every internal detail; it means ensuring that decision-makers and affected stakeholders receive the limitations material to their reliance.

Assurance as structured challenge

Assurance is the disciplined act of challenging a trust claim. It asks whether the evidence supports the conclusion and whether important counterevidence has been considered. Assurance may occur through peer review, independent testing, architecture review, red teaming, operational-readiness review, audit, or certification. The method should be proportionate to consequence and independent enough to resist confirmation bias.

Assurance should not become a centralized bottleneck. Mature organizations embed review and evidence generation into normal engineering workflow, then reserve deeper independent challenge for higher-risk decisions. This preserves speed while increasing the quality of consequential decisions.

5. Trust Across the Lifecycle

Trust cannot be inspected into a system at release. It develops—or erodes—throughout the lifecycle. Early decisions establish the problem definition, intended users, prohibited uses, risk tolerance, and accountable ownership. Requirements translate these into testable obligations. Architecture defines boundaries, failure containment, data flows, dependencies, and intervention. Implementation creates the behavior. Review and verification challenge the claims. Release controls determine whether the evidence is sufficient for the next operating context. Operations test the assumptions against reality. Stewardship decides when change, restriction, or retirement is necessary.

Design-time trust focuses on intent and structure. Teams identify stakeholders, context of use, potential impacts, hazards, misuse, security threats, data constraints, authority, and human oversight. ISO/IEC 42005 provides lifecycle guidance for AI system impact assessment and explicitly connects impact analysis with transparency, accountability, and trust [6]. The value of impact assessment is not the document itself; it is the design change and decision discipline the assessment produces.

Delivery-time trust focuses on controlled change. Source control, code review, automated checks, test environments, build integrity, approvals, release evidence, and separation of duties reduce the chance that an unreviewed or unsupported change reaches users. NIST's Secure Software Development Framework recommends integrating fundamental secure-development practices into any SDLC implementation [7]. The trust perspective broadens the objective from vulnerability reduction to justified reliance in the delivered system.

Runtime trust focuses on observed behavior. OpenTelemetry defines observability as understanding internal system state through outputs such as traces, metrics, and logs [8]. For intelligent systems, telemetry should also capture relevant prompts or instructions, retrieval and tool calls, model and configuration versions, policy decisions, escalations, refusals, human interventions, and outcome signals—subject to privacy and security

constraints. Runtime evidence allows teams to test whether the system continues to satisfy the assumptions that justified release.

Stewardship trust focuses on sustained fitness. Systems accumulate dependency risk, model drift, data changes, workarounds, exceptions, technical debt, and new uses. Trust can expire. Organizations need review triggers, recertification conditions, ownership continuity, deprecation criteria, and retirement plans. A system should not retain authority merely because it once passed a gate.

Lifecycle layer	Primary trust concern	Decision enabled
Design time	Purpose, impact, boundaries, architecture, ownership	Should this system or capability be built, and under what constraints?
Delivery time	Controlled change, review, validation, provenance, release evidence	Is this change supported strongly enough to enter the target environment?
Runtime	Observed behavior, reliability, misuse, drift, escalation, recovery	May the system continue operating at its current authority?
Stewardship	Fitness over time, accumulated risk, lessons, retirement	Should the system be changed, restricted, recertified, or retired?

6. Operational Reality Is the Final Arbiter

A system can be well designed, thoroughly reviewed, and still fail in production. Real users behave differently from test participants. Dependencies degrade. Data distributions shift. Attackers adapt. Workarounds emerge. Business processes change. Operational trust therefore depends on the organization's capacity to observe, interpret, respond, recover, and learn.

Reliability is a direct contributor to trust. Google SRE states the relationship plainly: if a system is not reliable, users will not trust it [9]. Reliability, however, is more than uptime. A service can be available while returning the wrong result, applying the wrong policy, or taking an inappropriate action. Service level indicators and objectives should reflect what users actually depend upon, including correctness, latency, freshness, completion, and safe degradation.

Resilience adds the ability to anticipate, withstand, recover from, and adapt to adverse conditions. NIST's cyber-resiliency guidance frames resilience as a means of sustaining system trustworthiness in the presence of attacks, compromises, and stress [10]. Recovery evidence—restore tests, failover exercises, rollback demonstrations, incident simulations, and revocation drills—is often more persuasive than claims that failure is unlikely.

Incident response is part of the trust relationship. Stakeholders judge not only whether a failure occurred, but whether it was detected, contained, communicated, corrected, and prevented from recurring. Honest incident communication can preserve justified trust; concealment or minimization can destroy it. Postmortems should convert operational experience into architecture, test, process, and governance improvements.

Organizations must also be prepared to reduce authority. When evidence deteriorates, a system may need to move from autonomous operation to approval-required mode, from broad deployment to a constrained population, or from live action to recommendation only. Progressive de-authorization is as important as progressive authorization.

7. Agentic Systems Change the Trust Boundary

Agentic systems make the trust problem more demanding because they can convert inference into action. An agent may decompose a goal, select tools, retrieve context, retain memory, coordinate with other agents, and act across multiple systems. Each capability expands the control surface and the potential blast radius.

Trust in an agent should never be inferred from fluent interaction or benchmark performance alone. The relevant questions concern authority: Which goals may it pursue? Which resources may it access? Which tools may it invoke? Which transactions may it execute? What state may it retain? When must it stop, ask, or escalate? Who can revoke its identity? How is its activity reconstructed after the fact?

OWASP's Top 10 for Agentic Applications identifies risks such as goal hijacking, tool misuse, identity and privilege abuse, memory poisoning, and cascading failures across autonomous workflows [11]. These risks show why conventional application security is necessary but insufficient. The system must also govern intent, context, memory, delegation, and inter-agent coordination.

Trustworthy agentic design uses bounded autonomy. Authority is narrow, explicit, time-limited where appropriate, and proportionate to consequence. Agents have distinct identities rather than borrowed human credentials. Tool access follows least privilege. High-impact actions require independent approval or additional evidence. Handoffs preserve context, assumptions, authority, and validation expectations. Memory has provenance, retention rules, and correction mechanisms. Runtime monitoring detects unusual action sequences and policy violations.

Most importantly, authority should be earned progressively. A new agent begins in assistive or simulated modes. It advances only when evidence demonstrates acceptable behavior under increasingly realistic conditions. Authority can expand by task, data domain, environment, transaction value, or operating window. It must also contract when incidents, drift, context changes, or unexplained behavior weaken the assurance basis.

Agentic Trust Principle

Do not ask whether the agent is trustworthy. Ask which actions it is authorized to take, under which conditions, based on which evidence, with which independent checks, and with what means of immediate containment.

8. Human Trust, Organizational Trust, and Accountability

Technology does not operate outside institutions. Trust in a system is inseparable from trust in the organization that designs, deploys, and governs it. Stakeholders rely on the organization to select appropriate purposes, disclose material limitations, respond to harm, protect data, maintain competence, and correct failures.

Human oversight must therefore be meaningful rather than ceremonial. A reviewer needs time, information, authority, competence, and a usable intervention path. An approval button does not create accountability when the human cannot understand the recommendation, inspect the evidence, or resist organizational pressure. In high-tempo systems, oversight may need sampling, independent monitoring, preapproved policy, or exception review rather than manual approval of every action.

Professional culture matters. Teams that punish the discovery of bad news create hidden risk. Teams that reward output volume without evidence create false confidence. Teams that separate builders from operational consequences weaken learning. Trustworthy organizations make uncertainty discussable, findings actionable, exceptions visible, and ownership durable.

Decision rights should be explicit. Product leaders own intended value and user impact. Engineering owns system design and implementation quality. Security, safety, privacy, legal, risk, and domain experts provide specialized challenge. Operations owns service capability and incident response. Executives own risk acceptance and resource allocation. These roles overlap, but accountability should not dissolve into consensus.

External trust may also require transparency beyond the engineering organization. Affected users may need understandable explanations, appeal paths, support, notice of automation, or disclosure of material

limitations. Regulators, customers, partners, and auditors may require evidence of process and outcome. The form should match the stakeholder, but the underlying record should remain coherent.

9. Trust Debt and the Failure Modes of Trust Theater

Trust debt accumulates when an organization relies on assumptions it cannot readily justify. Examples include undocumented data provenance, unexplained model behavior, unowned exceptions, untested recovery, shared credentials, missing telemetry, stale impact assessments, unresolved review findings, or production workflows that depend on informal human knowledge. Like technical debt, trust debt may remain invisible while conditions are stable and become expensive during change or crisis.

Trust theater occurs when visible signals substitute for substantive assurance. Common forms include ethics principles without engineering controls, review boards without evidence, model cards copied from templates, dashboards that omit consequential outcomes, human-in-the-loop labels without meaningful intervention, and compliance checklists that are disconnected from system behavior.

Another failure mode is trust inflation: language such as safe, secure, responsible, autonomous, or enterprise-ready is used more broadly than the evidence warrants. Trust inflation creates risk because users and downstream teams make decisions based on the implied assurance. Professional communication should use bounded claims and state the relevant conditions.

Overcontrol can also reduce trustworthiness. Excessive gates, duplicated documentation, centralized approvals, and risk processes disconnected from delivery encourage workarounds and stale evidence. The objective is not maximum governance; it is effective governance. Controls should be embedded where decisions and evidence are produced, automated when the rule is stable, and escalated when judgment is necessary.

Finally, organizations may measure trust in ways that invite gaming. Counts of reviews, tests, controls, explanations, or incidents do not establish quality. Metrics are useful when they illuminate the health of the trust system, but they become dangerous when treated as the trust result itself.

10. Measuring Trust Without Pretending It Is a Single Number

Trust is multidimensional and contextual; collapsing it into a single score obscures tradeoffs. A system can improve reliability while increasing privacy risk, or improve explainability while exposing sensitive information. A useful measurement model therefore connects each metric to a trust claim and a decision.

Leading indicators assess the capability to produce trustworthy outcomes: requirement clarity, unresolved high-severity risks, review depth, test relevance, provenance coverage, rollback success, telemetry completeness, exception age, and ownership continuity. Lagging indicators assess what occurred: user-impact incidents, harmful outcomes, security events, policy violations, recovery performance, appeals, and repeated failure modes.

Calibration measures whether confidence matches reality. Teams can compare predicted risk with observed outcomes, reviewer confidence with defect escape, model confidence with error, and authorization level with incident rate. Calibration is especially important where AI fluency encourages overreliance.

Trust metrics should be segmented by context, population, use case, version, environment, and authority level. Aggregates can conceal concentrated harm or fragile conditions. The measurement system should also preserve counterevidence and failed tests rather than reporting only pass rates.

The purpose of measurement is decision support. Metrics should help leaders determine whether to release, expand, restrict, investigate, remediate, or retire a capability. When a metric does not change a decision, it may be reporting overhead rather than assurance.

A five-level trust maturity model

Level	Operating characteristic	Trust posture
1. Asserted	Trust is expressed through principles, reputation, and project claims.	Reliance depends heavily on confidence and individual judgment.
2. Controlled	Selected risks have documented controls, reviews, and owners.	Trust is bounded but evidence remains fragmented.
3. Evidenced	Claims link to current, reviewable lifecycle evidence.	Reliance can be justified for defined contexts.
4. Operational	Telemetry, incidents, drift, exceptions, and recovery continuously inform authority.	Trust is actively calibrated and can contract or expand.
5. Adaptive	Governance, evidence, and authorization evolve systematically with consequence and learning.	Trust becomes a managed organizational capability.

11. An Enterprise Operating Model for Engineering Trust

Engineering trust at scale requires an operating model, not a collection of expert reviews. The organization needs common language for trust claims, risk tiers, evidence expectations, decision rights, exceptions, and lifecycle states. It also needs platforms that make the right work easier than the workaround.

At the team level, trust is built through explicit requirements, architecture decisions, repository workflow, peer review, automated verification, release evidence, observability, and operational ownership. At the platform level, organizations provide reusable identity, policy, logging, provenance, evaluation, deployment, and recovery capabilities. At the enterprise level, governance defines risk appetite, specialized review, escalation, external obligations, and accountability.

Federation is usually more effective than centralization. Central functions establish policy, shared controls, expertise, and independent challenge. Product and engineering teams remain responsible for evidence and outcomes. This preserves domain knowledge and delivery speed while reducing inconsistency.

Trust platforms should avoid becoming evidence warehouses disconnected from work. The strongest evidence is generated by normal engineering activity and linked to the decisions it supports. Repositories, CI/CD systems, identity platforms, observability systems, incident tools, and governance records should form a traceable evidence fabric rather than isolated reporting silos.

Leaders should fund trust as product and platform capability. Secure defaults, evaluation harnesses, policy enforcement, provenance, telemetry, approval workflows, and recovery automation reduce the marginal cost of trustworthy delivery across many teams. The alternative is repeated manual assurance and inconsistent local reinvention.

Consulting organizations and technology providers are increasingly focused on AI operating models, control planes, agent governance, platform engineering, and responsible scaling. The enduring opportunity is not to add more oversight around delivery, but to redesign delivery so that trust-relevant evidence and controls are native to the workflow.

12. The ETIS Position: Trust Must Be Engineered and Earned

Engineering Trustworthy Intelligent Systems (ETIS) treats trust as a lifecycle result, not a model attribute, product claim, or compliance label. The model is not the system. A demonstration is not operational proof.

Governance is architecture. Context is control. Everything important leaves evidence. Human beings remain accountable.

ETIS integrates repository-centered engineering, evidence-centered engineering, governance, review, operational readiness, context engineering, agentic authority, and stewardship into one coherent operating model. Each discipline contributes to trust, but none is sufficient alone. Repository structure without evidence quality becomes documentation hygiene. Evidence without governance does not determine who may decide. Governance without operational feedback becomes theater. Operations without architecture becomes reactive. Human oversight without authority becomes symbolic.

The ETIS trust model is claim-centered. Teams identify what reliance they seek, define the conditions and limits, design the architecture, generate evidence, subject the claim to appropriate challenge, authorize operation, observe outcomes, and revise authority as reality changes. Trust is earned in increments and remains conditional.

This approach aligns with established standards and practices while filling the connective gaps among them. NIST defines trustworthy AI characteristics and risk-management outcomes [1]. ISO/IEC 42001 defines an AI management system [3]. NIST SP 800-160 provides systems-security engineering principles [2]. SSDF and SLSA address secure development and provenance [5][7]. SRE and OpenTelemetry provide operational disciplines and evidence [8][9]. OWASP identifies agentic security risks [11]. ETIS connects these concerns to the engineering lifecycle, repository, decisions, evidence, and human accountability.

The result is not certainty. Complex systems will fail, intelligent systems will surprise, and operating environments will change. The professional objective is a system whose reliance is explicit, whose boundaries are enforceable, whose behavior is observable, whose evidence is challengeable, whose failures are containable, and whose owners are accountable. That is the practical meaning of engineering trust.

ETIS Doctrine

Trust is not granted by fluency, novelty, reputation, certification, or a successful demonstration. Trust is earned through disciplined engineering and retained only while evidence, operating conditions, and accountable stewardship continue to justify reliance.

Conclusion

Software and intelligent systems are becoming more capable, more connected, and more consequential. The language of trust will therefore become more common—but its casual use will become more dangerous. Organizations that treat trust as branding, principle statements, or final-stage review will accumulate hidden dependence on systems they do not fully understand and cannot reliably control.

A stronger approach begins with justified reliance. It specifies what is being trusted, by whom, for what purpose, within which conditions, and on the basis of which evidence. It treats trustworthiness as an emergent system property produced by architecture, controlled change, provenance, verification, governance, observability, resilience, meaningful human oversight, and stewardship.

The shift to agentic systems makes this discipline urgent. When software can plan and act, trust must be translated into identity, permissions, authority, context, intervention, and continuous evidence. Autonomy should grow only as assurance grows, and it must contract when the assurance basis weakens.

The central professional commitment is simple: do not ask stakeholders to trust what engineering cannot explain, bound, observe, challenge, recover, and own. Build systems that deserve the reliance placed upon them.

References

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

[2] National Institute of Standards and Technology. Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems, NIST SP 800-160 Vol. 1 Rev. 1, 2022. https://csrc.nist.gov/pubs/sp/800/160/v1/r1/final

[3] International Organization for Standardization. ISO/IEC 42001:2023, Artificial intelligence — Management system. https://www.iso.org/standard/42001

[4] Cybersecurity and Infrastructure Security Agency. Secure by Design. https://www.cisa.gov/securebydesign

[5] SLSA. Supply-chain Levels for Software Artifacts, Build Provenance specification. https://slsa.dev/spec/v1.2/build-provenance

[6] International Organization for Standardization. ISO/IEC 42005:2025, Artificial intelligence — AI system impact assessment. https://www.iso.org/standard/42005

[7] National Institute of Standards and Technology. Secure Software Development Framework (SSDF) Version 1.1, NIST SP 800-218, 2022. https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-218.pdf

[8] OpenTelemetry. Observability Primer and OpenTelemetry documentation. https://opentelemetry.io/docs/concepts/observability-primer/

[9] Google. Site Reliability Engineering: Reaching Beyond Your Walls; reliability and user trust. https://sre.google/workbook/reaching-beyond/

[10] National Institute of Standards and Technology. Developing Cyber-Resilient Systems: A Systems Security Engineering Approach, NIST SP 800-160 Vol. 2 Rev. 1, 2021. https://csrc.nist.gov/pubs/sp/800/160/v2/r1/final

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

[12] National Institute of Standards and Technology. Trustworthy and Responsible AI. https://www.nist.gov/trustworthy-and-responsible-ai

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

[14] National Institute of Standards and Technology. AI RMF Playbook. https://airc.nist.gov/airmf-resources/playbook/

[15] International Organization for Standardization. Artificial intelligence standards portfolio. https://www.iso.org/sectors/it-technologies/ai

[16] Google. Site Reliability Engineering: Testing for Reliability. https://sre.google/sre-book/testing-reliability/

[17] OpenTelemetry. What is OpenTelemetry? https://opentelemetry.io/docs/what-is-opentelemetry/

[18] CISA. Principles and Approaches for Security-by-Design and Default, 2023. https://www.cisa.gov/sites/default/files/2023-06/principles_approaches_for_security-by-design-default_508c.pdf

[19] SLSA. About SLSA and software supply-chain trust. https://slsa.dev/spec/v1.2/about

[20] OWASP GenAI Security Project. Agentic AI Threats and Mitigations. https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/

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 Trust: How Trust Emerges from Architecture, Evidence, Governance, and Operational Reality. ETIS White Paper Series, WP-011.