ETIS WHITE PAPER SERIES

WP-006

Engineering Governance

Turning Principles into Decision Rights, Enforceable Controls, and Accountable Change

CORE THESIS

Governance is not a committee surrounding engineering. It is the architecture of authority, evidence, and intervention through which an organization decides what may change, who may decide, what proof is required, and how consequences are owned.

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-006 | Version 1.0

Executive Abstract

Software organizations have always governed consequential change. Architecture boards approve exceptions, security teams define controls, product leaders allocate risk, release managers establish baselines, and operators decide whether a system is safe enough to run. Yet governance has often been treated as an external review layer: policies are written apart from engineering, approvals arrive late, evidence is assembled for an audit, and accountability becomes diffuse when the system behaves differently in production.

AI-assisted delivery and agentic systems make that model untenable. AI can accelerate requirements analysis, generate implementation, modify repositories, run tools, coordinate multi-step work, and increasingly act within operational workflows. The resulting risk is not limited to incorrect output. It includes incorrect authority: a system may use the wrong data, invoke the wrong tool, cross a permission boundary, make a high-consequence decision without meaningful review, or continue operating after its assumptions have changed. Standards and state-of-practice guidance are converging on lifecycle governance, impact assessment, continual monitoring, explicit accountability, and controls that are proportionate to risk. NIST frames governance as a cross-cutting function that informs mapping, measurement, and management of AI risk. ISO/IEC 42001 establishes a management-system approach, while ISO/IEC 42005 connects governance to system-level impact assessment. GitHub rulesets and CODEOWNERS illustrate how policy can be translated into enforceable workflow. OWASP, Google SAIF, Microsoft, IBM, and AWS are extending governance toward agentic authority, lifecycle telemetry, and operational control. [1-12]

This paper argues that engineering governance should be understood as a control system for responsible change. It establishes decision rights, risk tolerances, evidence obligations, review boundaries, exception paths, runtime intervention, and stewardship responsibilities. Effective governance is neither centralized bureaucracy nor unrestricted team autonomy. It is a federated model in which organizational policy is translated into engineering constraints, teams retain local judgment inside those constraints, and high-consequence decisions receive stronger challenge and proof.

Engineering Trustworthy Intelligent Systems (ETIS) operationalizes this model through repository-centered engineering, evidence-centered engineering, phase-gate review, explicit authority boundaries, release governance, runtime observability, and lifecycle stewardship. The objective is not to create more approvals. It is to make consequential decisions visible, proportionate, enforceable, reviewable, and accountable before operational reality makes the decision on the organization's behalf.

KEY IDEA

Good governance does not slow engineering by default. It removes ambiguity about authority, evidence, risk, and escalation so that routine change moves quickly and consequential change receives the scrutiny it deserves.

1. Governance at an Engineering Inflection Point

The word governance often evokes boards, policy documents, compliance reviews, and executive oversight. Those mechanisms remain necessary, but they are incomplete. Software systems are shaped every day by thousands of engineering decisions: which dependency to introduce, which data to retain, which service may call another, which failure can be tolerated, which model or tool may be used, which branch may be merged, which release may proceed, and which operational signal requires intervention. If governance is absent from those decisions, it is absent from the system.

The AI era increases both the number and consequence of these decisions. Generative systems are nondeterministic. Agentic systems can execute tools and change state. Models, prompts, retrieval settings,

policies, permissions, and evaluation data may all alter behavior without a conventional code change. Third-party models and services introduce external dependencies that evolve outside the organization's direct control. A system may pass a release review and still drift, encounter new attack patterns, or behave differently as context and users change.

This does not mean that every decision requires a committee. It means that an organization needs an explicit mechanism for distinguishing routine decisions from material ones, assigning accountable owners, defining evidence expectations, and preserving the ability to challenge, stop, reverse, or improve a decision. Governance becomes an engineering capability when those mechanisms are part of architecture and workflow rather than an after-the-fact administrative process.

NIST AI RMF makes governance cross-cutting rather than sequential: the Govern function informs Map, Measure, and Manage. The NIST Playbook further emphasizes that the framework is not a universal checklist; organizations must translate outcomes into actions suited to their context and risk tolerance. ISO/IEC 42001 similarly treats governance as a management system that must be established, implemented, maintained, and continually improved. These directions reinforce a central conclusion: governance is not a single gate. It is a continuous operating discipline. [1-4]

2. From Oversight to an Engineering Control System

Traditional governance often begins with policy and ends with attestation. Engineering governance begins with the decision and follows it through consequence. It asks six practical questions: What decision is being made? Who has the authority to make it? What evidence is required? Which independent challenge is necessary? What controls enforce the decision? What happens when the decision proves wrong?

Viewed this way, governance resembles a feedback control system. Policy defines acceptable boundaries. Architecture translates those boundaries into system constraints. Workflow ensures that material changes receive the appropriate review. Verification produces evidence. Release establishes an accountable baseline. Telemetry reveals actual behavior. Incidents, exceptions, and postmortems feed learning back into policy and design. Governance fails when any link is missing: a policy without enforcement, a control without evidence, a review without accountability, or monitoring without intervention.

The objective is controlled adaptability. Overly centralized governance creates queues, encourages teams to bypass controls, and substitutes procedural compliance for engineering judgment. Unrestricted autonomy produces inconsistency, hidden risk, and local optimization. A mature organization establishes paved roads for common work, mandatory controls for material risk, and clearly defined exception paths for cases that do not fit the standard pattern.

Governance layer	Primary question	Representative mechanisms	Evidence produced
Design-time governance	What may be built, for whom, under what constraints?	Requirements, impact assessment, architecture, threat modeling, data and authority boundaries	Approved scope, risk classification, ADRs, impact and security records
Delivery-time governance	What may change, who may approve it, and what proof is required?	Issues, protected branches, CODEOWNERS, pull requests, required checks, release gates, provenance	Review history, test and scan results, approvals, exceptions, release baseline
Runtime governance	Is the system behaving within policy and risk tolerance?	Identity, least privilege, policy enforcement, telemetry, guardrails, rate limits, human intervention, rollback	Behavioral evaluations, audit trails, alerts, interventions, incidents

Stewardship governance	Who owns the system after release, and how does it improve?	Operational ownership, recurring review, model and dependency monitoring, postmortems, retirement criteria	Residual-risk record, corrective actions, maturity plan, decommission evidence

3. Decision Rights Are the Foundation of Accountability

Organizations frequently say that humans remain accountable for AI. The statement is correct but incomplete. Accountability is credible only when decision rights are explicit. A named person cannot meaningfully own a consequence if authority is fragmented, evidence is unavailable, or the organization has not defined which decisions require human judgment.

Decision rights should be attached to categories of consequence rather than titles alone. Product owners may authorize business outcomes and acceptable user impact. Architects may approve structural boundaries and exceptions. Security and privacy owners may define protected data and prohibited actions. Engineering leads may accept implementation and operational risk within delegated limits. Release authorities may establish baselines. Operators may halt or degrade a service. Executives and governing bodies may set risk appetite, fund controls, and accept residual enterprise risk.

For agentic systems, the same discipline must extend to machine authority. The relevant question is not whether an agent is intelligent; it is what the agent is permitted to decide and do. Authority should be task-scoped, least-privileged, time-bounded where feasible, observable, and revocable. The greater the autonomy and potential impact, the stronger the required identity, evaluation, approval, monitoring, and intervention mechanisms. OWASP and Google SAIF both emphasize that agentic risk grows with permissions, tool access, memory, orchestration, and the ability to create real-world side effects. [9, 10, 13]

ETIS DOCTRINE

Authority must be designed before it is delegated. Human accountability cannot be added after a system has already been given consequential power.

4. Proportional Governance: Match Control to Consequence

A common governance failure is to treat every system and every change alike. Uniform control is usually either excessive for low-risk work or inadequate for high-risk work. Proportional governance begins by classifying consequence: who can be harmed, what data or resources can be affected, whether the action is reversible, how broadly failure can propagate, and how quickly the organization can detect and contain it.

Risk tiering should influence the full lifecycle. A low-consequence internal summarization tool may require disclosure, data handling rules, basic evaluation, and an accountable owner. A system that recommends employment, credit, health, safety, or disciplinary action may require formal impact assessment, independent review, subgroup evaluation, explainability, stronger change control, human decision authority, appeal mechanisms, and continuing monitoring. An agent that can change production state, transfer funds, modify records, or invoke privileged tools requires even stronger controls because the failure is an action, not merely an answer.

ISO/IEC 42005 formalizes the importance of impact assessment across the AI lifecycle, while Microsoft's Responsible AI Impact Assessment and Standard show how organizational principles can be translated into development requirements. NIST AI RMF profiles similarly allow organizations to tailor governance outcomes to use cases and risk tolerances. [2, 3, 6, 14]

Illustrative tier	Typical consequence	Minimum governance posture	Escalating controls
Tier 1: Assisted productivity	Low external impact; human directly reviews output	Named owner, acceptable-use rules, data controls, disclosure, basic validation	Periodic sampling and usage monitoring
Tier 2: Decision support	Output influences a material human decision	Impact analysis, documented limitations, evaluation, human decision rights, traceability	Independent review, subgroup analysis, appeal and override paths
Tier 3: Bounded execution	System invokes tools or changes state within constrained workflows	Agent identity, least privilege, approved tools, transactional limits, runtime telemetry, rollback	Dual control, sandboxing, behavioral red-team, continuous evaluation
Tier 4: High-consequence autonomy	System can create broad, difficult-to-reverse, regulated, safety, or financial impact	Executive risk acceptance, formal assurance, independent oversight, stringent runtime control and intervention	Continuous monitoring, mandatory human checkpoints, kill switch, external review where appropriate

5. The Repository and Delivery Platform as Governance Surface

Governance becomes reliable when it is encoded in the environments where engineering work occurs. Modern repositories and delivery platforms already support many governance mechanisms: protected branches, rulesets, required reviews, CODEOWNERS, signed commits, status checks, environment approvals, dependency policies, artifact attestations, and release records. These controls do not eliminate judgment. They make the required judgment visible and difficult to bypass silently.

GitHub rulesets can control interactions with branches and tags, require reviews, restrict deletion or force pushes, require signed commits, and enforce status checks. CODEOWNERS connects sensitive areas of a repository to responsible individuals or teams. Used well, these features translate organizational expectations into repeatable workflow: architecture files require architecture review; security-sensitive code requires security ownership; model, prompt, policy, or evaluation changes trigger specialized checks; production releases require an accountable baseline. [7, 8]

The repository is also where governance evidence can remain connected to the system. An impact assessment links to requirements and architecture. An exception links to the decision and compensating control. A pull request records challenge and approval. CI/CD records automated verification. A release record establishes what was authorized. Operational findings link back to defects, risks, tests, and future changes. This chain is far stronger than a disconnected approval spreadsheet because it preserves context, lineage, and the actual object being governed.

However, automation can create a false sense of safety. A green pipeline proves only that configured checks passed. It does not prove that the correct risks were identified, the right tests were selected, or a human understood the change. Engineering governance should therefore combine automated enforcement with explicit human judgment for material decisions.

6. Governing AI-Assisted and Agentic Engineering

AI is becoming a participant in the engineering process. Coding agents can inspect repositories, generate plans, modify files, execute commands, and prepare pull requests. The governance question is therefore broader than whether AI-generated code is allowed. Organizations must govern which agents may operate, what repositories and environments they may access, what tools they may invoke, how secrets and sensitive context are protected, what evidence they must produce, and which human must review the result.

A useful distinction is between assistance, coordination, execution, and operation. Assistance produces suggestions under direct human control. Coordination organizes tasks and context. Execution performs bounded multi-step engineering work. Operation acts within a live environment. Each level transfers more authority and therefore requires stronger controls. Agent identity and authorization are becoming a distinct standards concern, and OWASP's agentic guidance highlights threats such as goal hijacking, tool misuse, excessive agency, memory poisoning, and cascading failures. [9-11, 15]

Governance should also address model and vendor independence. Organizations increasingly use multiple foundation models, coding agents, SaaS services, and open-source components. A governance model tied to one provider will not scale. The control plane should maintain an inventory of use cases, models, agents, data sources, tools, policies, owners, evaluations, and operational status across platforms. IBM watsonx.governance, Microsoft's Responsible AI Standard and transparency practices, AWS governance-by-design guidance, and Google SAIF all point toward lifecycle-wide governance rather than one-time model approval. [5, 6, 12, 16-19]

ETIS treats AI use as provenance rather than confession. The relevant evidence is what tool or agent was used, what task it performed, what context and authority it received, what output was accepted or rejected, what human changes occurred, how the result was verified, and who owns the consequence.

7. Exceptions, Escalation, and Meaningful Human Oversight

No governance system can anticipate every engineering situation. Exceptions are therefore not evidence that governance failed; unmanaged exceptions are. A mature exception process records the policy being bypassed, the business or engineering justification, the accountable approver, the compensating controls, the duration, the affected baseline, and the condition for closure.

Exception paths should be easier to use than informal circumvention. When teams believe a control is unnecessary or harmful, the organization should be able to examine whether the control is incorrectly designed, whether the case is genuinely unusual, or whether risk tolerance has changed. Repeated exceptions are evidence that the standard path requires redesign.

Human oversight must also be engineered. A nominal approval box is not meaningful if the reviewer lacks time, information, authority, or the ability to intervene. Effective oversight requires understandable evidence, clear decision options, visible uncertainty, and practical mechanisms to pause, override, rollback, or escalate. For high-consequence systems, the organization should test intervention paths just as it tests normal functionality.

The strongest question is not 'Was a human in the loop?' It is 'Did the human have the competence, context, authority, and time needed to change the outcome?'

REVIEW PRINCIPLE

A review is not governance because it occurred. It becomes governance when the reviewer can challenge the claim, access the evidence, understand the consequence, and stop or redirect the decision.

8. Governance Evidence, Metrics, and Assurance

Governance cannot be evaluated solely by counting policies, reviews, or approvals. Those measures reveal activity, not effectiveness. Useful governance metrics connect control to outcome: how quickly material risks are identified, how often required evidence is missing, how many exceptions remain open, whether high-risk changes receive independent challenge, how frequently runtime limits are triggered, whether incidents reveal previously undocumented assumptions, and whether corrective actions close the underlying control gap.

The evidence model should distinguish design evidence, process evidence, behavioral evidence, and outcome evidence. Design evidence shows intended boundaries. Process evidence shows that required workflow occurred. Behavioral evidence shows how the system performed under evaluation and operation. Outcome evidence shows whether the system delivered value without unacceptable harm or instability. Governance is weak when it relies on only one category.

ISO/IEC 42006 extends the standards landscape toward certification bodies for AI management systems, reinforcing that assurance will increasingly examine whether governance processes are credible and repeatable. NIST, ISO, and vendor frameworks all emphasize continual improvement. Assurance therefore should not be a snapshot. It should assess whether the organization detects change, learns from operation, and updates controls. [1-6, 20]

The proper goal is minimum sufficient evidence: enough to support a responsible decision, proportionate to consequence, without turning evidence production into performative bureaucracy.

9. The Engineering Governance Operating Model

Engineering governance requires an operating model, not only a policy library. The operating model defines how enterprise risk appetite becomes product and engineering constraints; how teams obtain approved platforms and patterns; how specialist expertise is engaged; how exceptions are resolved; and how operational learning changes future decisions.

A practical model is federated. A central governance function establishes principles, taxonomies, minimum controls, shared tooling, and reporting. Platform and security teams encode reusable controls into paved roads. Product and engineering teams own system decisions and evidence. Independent reviewers challenge high-consequence claims. Operations owns runtime intervention and learning. Executives retain accountability for risk appetite, investment, and unresolved residual risk.

This model avoids two extremes. In a purely centralized system, governance becomes a bottleneck that lacks product context. In a purely decentralized system, every team reinvents policy and evidence. Federation creates shared boundaries with local engineering judgment.

Leadership incentives matter. If leaders reward feature volume while treating review, testing, documentation, and operational readiness as delay, teams will optimize around governance. If leaders use evidence to make investment, release, and risk decisions, governance becomes part of delivery quality rather than an external tax.

Role	Primary governance responsibility	Decisions that should remain explicit
Executive / governing body	Risk appetite, strategic outcomes, unresolved enterprise risk	Which consequences the organization is willing to accept and fund
Product and business owner	Purpose, users, value, impact, human decision model	What the system should achieve and where automation is appropriate
Engineering and architecture	System boundaries, implementation, change control, technical risk	How the system is structured and what evidence justifies change
Security, privacy, legal, compliance	Protected interests, prohibited actions, regulatory and contractual constraints	Which boundaries are mandatory and which exceptions require escalation
Independent review / assurance	Challenge of consequential claims and control effectiveness	Whether evidence is sufficient for a decision
Operations / SRE	Runtime limits, intervention, incident response, recovery, learning	When to degrade, stop, rollback, or retire the system
Steward / system owner	Lifecycle ownership, recurring review, decommissioning	Who remains accountable after the project team moves on

10. Common Governance Failure Modes

Governance theater occurs when artifacts exist but do not influence decisions. An impact assessment is completed after architecture is fixed. A review board approves a slide deck without inspecting evidence. An AI inventory lists models but omits agents, tools, data, and operational status. A policy requires human oversight but does not define decision authority or intervention. A risk is accepted without a named owner or expiration date.

Checklist governance is another failure mode. Checklists are useful memory aids, but they become dangerous when completion substitutes for reasoning. NIST explicitly cautions that AI RMF actions are not an ordered universal checklist. The appropriate controls depend on context, consequence, and organizational capacity. [1, 2]

Governance fragmentation occurs when architecture, security, model risk, data governance, compliance, release management, and operations maintain separate records that do not connect. Each function may be locally correct while the system remains globally ungoverned. Repository-centered evidence and shared system inventories can reduce this fragmentation.

Finally, governance can fail through excessive caution. If low-risk experimentation requires the same process as high-consequence deployment, teams will avoid the official path or the organization will lose learning opportunities. Proportionality, sandboxes, synthetic data, bounded pilots, and staged authority allow innovation while preserving control.

11. A Maturity Path for Engineering Governance

Organizations do not become governed by publishing a policy. Capability develops through repeated decisions, reusable controls, operational feedback, and cultural reinforcement. A useful maturity path begins with visibility and progresses toward adaptive lifecycle governance.

The progression is not purely technical. Tools can automate approvals and collect evidence, but maturity depends on whether decision rights are understood, reviewers exercise judgment, leaders act on residual risk, and operational learning changes policy.

Level	Governance posture	Characteristic practices	Primary limitation
1. Reactive oversight	Governance appears after a problem or before an audit	Informal approvals, scattered policy, project-specific records	Risk and ownership remain largely invisible
2. Documented control	Policies and required reviews are defined	Templates, inventories, review boards, basic risk classification	Controls rely heavily on manual interpretation
3. Embedded governance	Controls operate inside architecture and delivery workflow	Rulesets, CODEOWNERS, required checks, impact assessment, evidence-linked release gates	Runtime governance may remain weak
4. Lifecycle governance	Design, delivery, runtime, and stewardship are connected	Continuous evaluation, agent identity, telemetry, exceptions, intervention, postmortem feedback	Cross-portfolio consistency and outcome measurement remain difficult
5. Adaptive governance	Controls evolve with evidence, consequence, and system change	Risk-based automation, reusable control plane, independent assurance, policy feedback, retirement governance	Requires strong culture, platform capability, and executive accountability

12. The ETIS Perspective

Engineering Trustworthy Intelligent Systems treats governance as architecture. This doctrine does not mean that governance is only a technical design concern. It means that governance becomes credible when authority, constraints, evidence, and intervention are designed into the sociotechnical system.

ETIS connects governance to the full lifecycle. Requirements define intended outcomes, affected stakeholders, constraints, and prohibited behavior. Architecture establishes trust boundaries, data ownership, context sources, human decision points, and agent authority. Repository workflow preserves decisions, reviews, AI provenance, tests, exceptions, and release baselines. Verification challenges both correctness and behavior. Release governance decides whether evidence is sufficient. Operations provides telemetry, intervention, incident response, and recovery. Stewardship keeps ownership and residual risk visible after deployment.

Several ETIS doctrines reinforce this model. Everything important leaves evidence. The model is not the system. AI proposes; engineers verify. Context is control. A demo is not operational proof. Trust must be designed, demonstrated, operated, and stewarded. Engineering governance is the mechanism that makes those doctrines actionable rather than aspirational.

ETIS does not replace NIST, ISO, OWASP, GitHub, DORA, SRE, or vendor governance platforms. It provides an implementation-oriented synthesis: a way to connect organizational expectations to engineering stages, repository evidence, review obligations, operational controls, and accountable lifecycle decisions.

ETIS POSITION

Governance should be strongest where consequences are greatest, fastest where evidence is routine, and continuously informed by how the system actually behaves.

13. Strategic Implications

For executives, engineering governance is an investment in scalable decision quality. The relevant question is not how many AI pilots exist, but whether the organization can repeatedly move from opportunity to controlled operation without losing accountability. Shared platforms, inventories, evidence models, and risk-based controls create leverage across portfolios.

For engineering leaders, governance should be designed as part of the delivery system. Teams need clear decision rights, paved roads, policy-as-workflow, independent challenge for consequential changes, and operational feedback. The greatest governance improvement may be reducing ambiguity rather than adding approval.

For architects and engineers, governance becomes part of professional design. Data access, identity, agent permissions, observability, rollback, and human intervention are not administrative afterthoughts. They are system properties.

For educators, governance should be taught through engineering decisions and evidence rather than abstract principles alone. Students should practice impact analysis, risk classification, architecture review, AI-use provenance, release defense, exception handling, and postmortem learning.

Conclusion

The future of engineering governance will not be defined by the size of a policy manual or the number of review committees. It will be defined by whether organizations can translate purpose and risk appetite into explicit authority, enforceable constraints, reviewable evidence, controlled release, observable behavior, and accountable intervention.

AI-assisted and agentic systems make that capability urgent. As software gains greater ability to generate, decide, coordinate, and act, governance must move closer to the work and deeper into the architecture. The organization must know which decisions belong to humans, which tasks may be delegated, what evidence justifies authority, and how the system can be stopped or corrected when assumptions fail.

Engineering governance is therefore not the opposite of speed. Properly designed, it is what allows speed to scale without turning uncertainty into unmanaged consequence. It creates the conditions under which teams can innovate confidently, reviewers can challenge meaningfully, leaders can accept risk explicitly, and operators can maintain trust after release.

The central principle is simple: consequential change should never be easier to execute than it is to understand, review, and own.

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] NIST AI Resource Center. AI RMF Core and Playbook: Govern, Map, Measure, and Manage. 2026. https://airc.nist.gov/airmf-resources/airmf/

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

[4] International Organization for Standardization. ISO/IEC 38507:2022, Governance implications of the use of artificial intelligence by organizations. https://www.iso.org/standard/56641.html

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

[6] Microsoft. Responsible AI Standard v2: General Requirements. 2022. https://www.microsoft.com/en-us/ai/tools-practices

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

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

[9] OWASP GenAI Security Project. OWASP Top 10 for Agentic Applications for 2026. 2025. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/

[10] OWASP GenAI Security Project. Agentic AI Threats and Mitigations. 2025. https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/

[11] OWASP GenAI Security Project. State of Agentic AI Security and Governance 2.01. 2026. https://genai.owasp.org/resource/state-of-agentic-ai-security-and-governance/

[12] Google. Secure AI Framework (SAIF) and Governance Controls. 2026. https://saif.google/

[13] Google. SAIF 2.0 Focus on Agents. 2026. https://saif.google/focus-on-agents

[14] Microsoft. Responsible AI Impact Assessment Template. 2022. https://www.microsoft.com/en-us/ai/tools-practices

[15] NIST National Cybersecurity Center of Excellence. Software and AI Agent Identity and Authorization. 2026.

[16] IBM. watsonx.governance: AI Governance Across the Lifecycle. 2026. https://www.ibm.com/products/watsonx-governance

[17] IBM. Agentic AI Governance, Evaluation and Lifecycle. 2026. https://www.ibm.com/new/announcements/agentic-ai-governance-evaluation-and-lifecycle

[18] Amazon Web Services. Governance by Design: The Essential Guide for Successful AI Scaling. 2025. https://aws.amazon.com/blogs/machine-learning/governance-by-design-the-essential-guide-for-successful-ai-scaling/

[19] Microsoft. Responsible AI Transparency Report 2025. https://www.microsoft.com/en-us/corporate-responsibility/responsible-ai-transparency-report/

[20] International Organization for Standardization. ISO/IEC 42006:2025, Requirements for bodies providing audit and certification of AI management systems. https://www.iso.org/standard/42006

[21] NIST. 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
[22] NIST. Secure Software Development Framework (SSDF), SP 800-218. 2022. https://csrc.nist.gov/projects/ssdf
[23] Google Cloud DORA. State of AI-Assisted Software Development 2025. https://dora.dev/dora-report-2025/
[24] O'Connell, William T. Engineering Trustworthy Software in the AI Era. ETIS White Paper Series, WP-001, 2026.
[25] O'Connell, William T. Repository-Centered Engineering. ETIS White Paper Series, WP-002, 2026.
[26] O'Connell, William T. Engineering Evidence. ETIS White Paper Series, WP-003, 2026.
[27] O'Connell, William T. Engineering Agentic Software Systems. ETIS White Paper Series, WP-004, 2026.
[28] O'Connell, William T. Engineering Education in the AI Era. ETIS White Paper Series, WP-005, 2026.

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 Governance: Turning Principles into Decision Rights, Enforceable Controls, and Accountable Change. ETIS White Paper Series, WP-006, Version 1.0.*