ETIS EXECUTIVE BRIEF SERIES

Why AI Changes Software Governance

From Policy Oversight to an Engineering Control System

William T. O'Connell, Ph.D.

Former IBM Distinguished Engineer and CTO, IBM Consulting Quality and Delivery

Adjunct Professor of Computer Science, Loyola University Chicago

Creator, Engineering Trustworthy Intelligent Systems (ETIS)

ETIS Executive Brief EB-001 | Version 1.0 | July 2026

Executive Thesis

AI changes governance because it changes the speed, scale, and locus of engineering decision-making. Governance can no longer operate mainly through periodic review, policy statements, and committees. It must become an embedded control system that shapes authority, context, evidence, release, runtime behavior, and accountability as work occurs.

Audience: Chief technology officers, chief information officers, chief digital officers, engineering executives, enterprise architects, risk leaders, and board-level technology committees.

Executive Summary

Artificial intelligence is rapidly becoming part of the software delivery system itself. Developers use AI to generate code, tests, designs, documentation, and operational artifacts. Coding agents can accept issues, inspect repositories, execute commands, change multiple files, run tests, and submit pull requests. Enterprise platforms are introducing agent control planes, custom-agent standards, and policy controls that govern which agents may act, where, and under whose authority [4-7]. The operating question for technology leaders is therefore no longer whether employees use AI. It is how consequential AI-assisted and agentic work is authorized, constrained, evidenced, and owned.

Traditional governance assumes that material decisions occur at visible gates: architecture review, security approval, model review, change approval, or release authorization. AI moves many decisions earlier, deeper, and faster. Requirements are interpreted by tools; design alternatives are generated in minutes; implementation may be delegated; context files shape agent behavior; models and prompts change independently of application code; runtime systems may select tools and act. A governance system that sees only the final release is governing too late.

The correct response is not to add more committees or require more documentation. NIST makes governance a cross-cutting function across mapping, measurement, and management of AI risk, and ISO/IEC 42001 defines an organization-wide management system for responsible AI development and use [1-3]. Those frameworks establish the obligation. The executive challenge is operationalization: translating principles and risk appetite into decision rights, repository rules, identity and permissions, evidence requirements, release controls, runtime policy, intervention mechanisms, and accountable ownership.

This brief presents an executive operating model for that transition. Its central recommendation is to treat governance as architecture: a designed system of authority, evidence, and intervention embedded in the software lifecycle and delivery platform. This model enables speed by automating routine controls, concentrating human judgment at consequential boundaries, and preserving a defensible record of why the organization trusted a change or delegated an action.

The Board-Level Question

Can management demonstrate—not merely assert—who or what was authorized to make a consequential technology change, what evidence justified it, which controls bounded it, and how the organization would detect and contain failure?

1. The Governance Problem Has Moved

For most enterprises, technology governance was designed around human-led projects, periodic releases, and relatively stable system boundaries. Policy was written centrally, interpreted by teams, checked at stage gates, and audited after implementation. That model was already strained by cloud platforms, DevOps, infrastructure as code, continuous delivery, and third-party software supply chains. AI intensifies the mismatch.

The first change is velocity. AI reduces the cost of producing candidate changes. More requirements, designs, code, tests, and configuration can be generated in less time. DORA's 2025 research characterizes AI as an amplifier: it magnifies the strengths of effective organizations and the dysfunctions of weak ones [8][9]. Where architecture, product focus, testing, review, and platform capability are strong, AI can increase flow. Where they are weak, AI creates downstream disorder faster.

The second change is decision distribution. Engineering judgment is increasingly encoded in prompts, repository instructions, agent skills, policy files, model routing, retrieval configuration, tool descriptions, and automated workflows. These artifacts may materially change behavior without passing through the same review path as application code. Governance must therefore recognize them as controlled engineering assets.

The third change is authority. An assistant suggests; an agent acts. When an agent can write code, alter infrastructure, call APIs, update records, or coordinate other agents, AI adoption becomes a transfer of authority. Every transfer creates obligations for identity, least privilege, task boundaries, approval, evidence, observation, intervention, and accountability.

The fourth change is runtime variability. Intelligent systems may produce different outputs or action sequences under similar conditions. Pre-release approval remains necessary but cannot prove all operational behavior. Governance must extend into telemetry, behavioral evaluation, incident response, authority contraction, and stewardship.

Executive Insight

AI governance is not a separate governance domain surrounding software engineering. It is becoming part of software governance itself.

2. Governance Must Shift from Oversight to Control

Oversight observes and reviews. Control shapes what may happen. Mature governance requires both, but many organizations overinvest in oversight because it is visible: committees, policies, inventories, attestations, and dashboards. The harder work is control design.

NIST's AI RMF organizes risk management through Govern, Map, Measure, and Manage, with governance designed as the cross-cutting foundation for the other functions [1][2]. ISO/IEC 42001 similarly requires leadership, policy, risk management, lifecycle controls, performance evaluation, monitoring, and continual improvement [3]. These frameworks imply that governance should influence decisions continuously rather than appear only as a final approval.

In an engineering control system, policy is translated into enforceable mechanisms. Risk tiers determine required evidence and decision authority. Identity systems distinguish human and non-human actors. Repository rules prevent unreviewed changes from entering protected baselines. Required checks validate security, quality, provenance, and policy. Deployment controls establish who may release and under what conditions. Runtime policy limits tool use, data access, transaction scope, and delegation. Telemetry reveals whether behavior remains inside the authorized envelope.

Human review remains essential, but it becomes targeted. Deterministic rules should be automated. Experts should focus on ambiguity, tradeoffs, exceptions, residual risk, and consequence. This reduces friction because teams no longer wait for committees to verify routine obligations that the platform can enforce continuously.

Governance mode	Primary mechanism	Typical weakness
Policy oversight	Policies, committees, questionnaires, periodic audit	Controls may be interpreted inconsistently and applied after design decisions are fixed.
Gate-based assurance	Architecture, security, model, and release reviews	Useful for consequential decisions, but slow and incomplete when change is continuous.
Engineering control system	Decision rights, repository workflow, automated evidence, runtime policy, intervention	Requires platform investment, ownership clarity, and disciplined operating design.

3. Five Executive Design Decisions

Technology leaders do not need to design every control, but they must make five enterprise decisions explicit. Without them, teams will improvise inconsistent governance or vendors will define it by default.

Decision 1: What authority may be delegated?

Classify AI participation by authority, not by product label. Assistance, coordination, bounded execution, and live operation require progressively stronger identity, evidence, monitoring, and approval. The organization should define prohibited actions, mandatory human decisions, reversible boundaries, transaction limits, data restrictions, and conditions for escalation.

Decision 2: Which sources and instructions are authoritative?

Agent behavior is shaped by context. Requirements, architecture, policies, repository instructions, retrieval sources, tool descriptions, and memory must have owners, versioning, review, and freshness rules. Context is part of the control plane; unmanaged context is unmanaged behavior.

Decision 3: What evidence is required for consequential change?

Evidence should be risk-proportionate and generated by normal workflow. Material changes should preserve intent, producer identity, affected assets, review, tests, security results, provenance, known limitations, approval, and rollback. AI-generated evidence cannot validate itself; accountable humans or independent controls must accept the claim.

Decision 4: Where must intervention remain possible?

The architecture must support pause, deny, revoke, quarantine, rollback, degrade, or require approval. “Human in the loop” is not meaningful unless the human has information, time, competence, authority, and a usable intervention path.

Decision 5: Who owns the consequence?

Accountability must survive automation. Product leaders own intended value and user impact; engineering owns system integrity; security, privacy, legal, risk, and domain functions provide specialized challenge; operations owns runtime capability; executives accept residual enterprise risk. Responsibility cannot be assigned to the model or diluted across a committee.

Governance Principle

Do not approve “AI” in the abstract. Authorize a defined system, use, authority envelope, evidence basis, owner, and operating condition.

4. The Repository and Platform Become Governance Surfaces

As engineering becomes AI-assisted and agentic, the repository and delivery platform become central governance surfaces. GitHub’s current direction illustrates the change: coding agents work from issues and submit pull requests; enterprise controls can standardize custom agents, protect agent-definition paths, delegate AI-policy administration, and govern how agents interact with repositories [4-7].

This is strategically significant. Governance can be encoded close to the work through protected branches, required reviews, CODEOWNERS, rulesets, signed commits, policy checks, test gates, provenance, release controls, and environment protections. Repository instructions can define “always do,” “ask first,” and “never do” boundaries for agents [10]. The same system that improves developer flow can make governance enforceable and observable.

The repository should not become the storage location for every enterprise record. It should become the engineering system of record: the connected chain needed to understand why work was authorized, what changed, who or what produced it, how it was challenged, what was released, and what happened in operation.

Platform engineering is therefore a governance investment. DORA notes that individual AI productivity gains are frequently lost to downstream disorder in testing, security review, and deployment when platform capability is weak [11]. A well-designed internal platform turns enterprise policy into reusable paved roads, lowers the marginal cost of compliant delivery, and produces evidence automatically.

CTO Implication

The most scalable AI-governance program may not sit in a governance office. It may be the engineering platform that makes authorized, evidenced, and recoverable change the default path.

5. Runtime Governance Becomes Mandatory

Classical software governance focuses heavily on design and release because deterministic systems are expected to execute predefined logic. Intelligent and agentic systems can select among alternatives at runtime, use dynamic context, invoke tools, retry, delegate, and retain state. Governance must therefore continue after deployment.

Runtime governance begins with observability. Leaders need assurance that the organization can reconstruct which model, policy, context, identity, tool, permission, and workflow produced a consequential outcome. Telemetry should capture policy decisions, tool calls, escalation, human intervention, abnormal action sequences, and business outcomes while respecting privacy and security.

It also requires authority management. Credentials should be attributable and least-privileged. Permissions should be task-, purpose-, environment-, and time-aware where feasible. High-consequence actions may require deterministic checks, dual control, or approval. Agents should not inherit the full authority of the invoking human merely because integration is easier.

Finally, governance must be able to contract authority. When incidents, drift, context changes, abnormal behavior, or weakened evidence appear, the system should move to a lower-authority mode: from execution to recommendation, from broad operation to a constrained population, or from automatic action to approval required. Revocation is a core governance capability, not a failure of adoption.

6. The Operating Model: Federated Accountability, Central Enablement

Centralized AI governance does not scale when every product decision, model update, or agent workflow requires a specialist committee. Fully decentralized governance does not scale when each team invents its own definitions, evidence, and controls. The appropriate model is federated.

Enterprise leadership defines risk appetite, prohibited uses, authority classes, minimum evidence, exception policy, and board reporting. Central technology, security, risk, data, legal, privacy, and responsible-AI functions provide standards, shared controls, expertise, and independent challenge. Engineering platforms implement reusable controls and evidence generation. Product teams remain accountable for the design, evidence, operation, and outcomes of their systems.

This model preserves domain knowledge and speed while reducing inconsistency. It also changes the role of governance specialists: from approving every decision to designing the system within which good decisions are made, monitoring effectiveness, and intervening where consequence or uncertainty warrants deeper review.

Exceptions should be treated as governed engineering decisions. Each should have a named owner, rationale, compensating control, expiration or review trigger, and visible status. Permanent undocumented exceptions are trust debt.

Layer	Accountability	Executive evidence
Enterprise	Risk appetite, prohibited uses, authority classes, policy, funding	Portfolio risk view, major exceptions, incidents, capability maturity, unresolved systemic gaps
Platform	Reusable controls, identity, policy enforcement, evidence, observability	Adoption, bypass rates, control effectiveness, time-to-compliant-delivery, intervention readiness
Product/System	Purpose, architecture, data, context, validation, release, operation	System evidence package, owner, residual risk, runtime outcomes, remediation
Independent assurance	Challenge, audit, red team, certification where appropriate	Findings, disposition quality, repeat failures, independence and coverage

7. What the CTO Should Measure

Technology leaders should avoid reducing governance to a single trust score or a count of policies completed. Metrics should show whether the governance system improves decisions and outcomes.

Useful leading indicators include the percentage of consequential systems with named owners and authority models; provenance coverage; protected-change coverage; unresolved high-severity findings; exception age; rollback and revocation test success; evaluation coverage tied to intended use; and telemetry completeness. Useful outcome indicators include change instability, security and policy incidents, harmful or incorrect actions, recovery time, repeated root causes, and user-impact measures.

Metrics should also reveal friction. Excessive approval delay, duplicated evidence, manual re-entry, and high bypass rates indicate that governance is not integrated with engineering. The objective is not maximum control activity. It is minimum unmanaged consequence.

DORA cautions against using raw AI consumption as a performance metric; “tokenmaxxing” can reward activity rather than value and create perverse incentives [12]. Executive measures should focus on customer outcomes, delivery performance, quality, stability, risk, and learning.

Measurement Rule

Measure whether governance changes decisions, constrains authority, improves evidence, reduces unmanaged risk, and accelerates safe delivery—not how many policies, reviews, or AI tokens the organization produces.

8. A 90-Day Executive Agenda

The transition does not require a multi-year governance transformation before value is realized. A focused ninety-day agenda can establish control over the highest-consequence areas while creating the architecture for scale.

First, inventory material AI-assisted and agentic uses by authority and consequence—not merely by vendor or model. Identify where AI can change code, infrastructure, records, communications, decisions, or live operations. Assign accountable owners.

Second, define a small enterprise authority model: assist, coordinate, execute, and operate. For each level, specify identity, permission, evidence, approval, telemetry, and intervention expectations. This creates a common language across technology, risk, and business leadership.

Third, select one or two engineering value streams and embed governance into the repository and platform. Protect critical paths, require appropriate review and checks, preserve AI provenance, and produce a release evidence package automatically. Use the pilot to remove manual duplication, not add it.

Fourth, establish runtime visibility and revocation for one consequential intelligent workflow. Demonstrate that the organization can reconstruct an action, identify the responsible actor and context, pause or revoke authority, and recover safely.

Fifth, define executive reporting around systems, authority, evidence, exceptions, incidents, and capability—not tool licenses or adoption percentages. The board should see whether the organization is expanding AI authority faster than its control and assurance capability.

Conclusion

AI changes software governance because it changes who or what can make engineering decisions, how quickly those decisions become change, and where behavior is determined. A governance model designed mainly around policy publication and periodic approval will not keep pace.

The answer is not heavier bureaucracy. It is a better engineered system: explicit decision rights, bounded authority, authoritative context, repository-centered controls, continuous evidence, targeted human challenge, runtime observability, intervention, and accountable stewardship.

Technology leaders should view this as an operating-model and platform challenge. Organizations that embed governance into architecture and workflow can move faster because teams know the rules, routine controls are automated, and consequential decisions receive better evidence. Organizations that keep governance outside engineering will experience either unmanaged risk or growing friction.

The strategic objective is clear: expand AI capability only as fast as the organization can explain, constrain, verify, observe, interrupt, and own it.

Final Executive Position

Governance is not the committee surrounding AI-enabled engineering. Governance is the architecture of authority, evidence, and intervention through which the enterprise turns accelerated capability into accountable change.

References and Executive Sources

[1] National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0), 2023. https://www.nist.gov/itl/ai-risk-management-framework

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

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

[4] GitHub. GitHub Copilot: Meet the New Coding Agent, 2025. https://github.blog/news-insights/product-news/github-copilot-meet-the-new-coding-agent/

[5] GitHub. Enterprise AI Controls and the Agent Control Plane, 2025. https://github.blog/changelog/2025-10-28-enterprise-ai-controls-the-agent-control-plane-are-in-public-preview/

[6] GitHub. Delegate AI Controls Management, 2025. https://github.blog/changelog/2025-11-03-delegate-ai-controls-management-to-members-of-your-enterprise/

[7] GitHub. Introducing Agent HQ, 2025. https://github.blog/news-insights/company-news/welcome-home-agents/

[8] DORA. State of AI-assisted Software Development 2025. https://dora.dev/dora-report-2025/

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

[10] GitHub. How to Write a Great AGENTS.md: Lessons from More Than 2,500 Repositories, 2025. https://github.blog/ai-and-ml/github-copilot/how-to-write-a-great-agents-md-lessons-from-over-2500-repositories/

[11] DORA. Platform Engineering Capability and the AI Angle, 2026. https://dora.dev/capabilities/platform-engineering/

[12] DORA. Finding Balance in the Era of Tokenmaxxing, 2026. https://dora.dev/insights/

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

[14] International Organization for Standardization. ISO/IEC 38507:2022, Governance implications of the use of AI by organizations. https://committee.iso.org/committee/6794475/x/catalogue/

[15] William T. O'Connell. Engineering Governance. ETIS White Paper WP-006, 2026. https://etisframework.org

About the Author

William T. O'Connell, Ph.D., is the creator of Engineering Trustworthy Intelligent Systems (ETIS), an adjunct professor of computer science at Loyola University Chicago, and a former IBM Distinguished Engineer and CTO for IBM Consulting Quality and Delivery. His career spans four decades of software engineering, enterprise architecture, product and platform development, quality, governance, and large-scale client delivery.

His work focuses on how technology organizations can use artificial intelligence to increase capability without surrendering engineering discipline, operational responsibility, governance, evidence, or human accountability.