Why AI Software Delivery Needs Its Own Governance Layer

Preserving the operating brief, evidence, and human authority across models, tools, and teams

Sam Moore

Founder and CEO, Ludian

August 2026

ludian.ai

> **Working-paper disclosure.** This paper combines verified academic, standards, regulatory, institutional, and industry research; original Ludian synthesis; and a factual description of Ludian V1. Recent preprints are identified as preprints. Product descriptions distinguish Available, Preview, Planned, and future design directions. The paper does not claim that Ludian invented the broader category or that current V1 implements complete enterprise governance.

Executive Thesis

Software delivery has acquired a new class of participant: a probabilistic system that can interpret requirements, generate code, invoke tools, run tests, inspect repositories, and propose changes across a delivery path. That system can move faster than the organizational context meant to govern it. When the brief lives in a chat, the code lives in a repository, the evidence lives in several tools, and the approval lives in a person's head, an organization can see the output without being able to reconstruct what was intended, which constraints governed the work, or why it was allowed to move forward. That is not only a model-risk problem. It is a governance gap inside software delivery.

Ludian is the governance layer for AI software delivery - preserving intent and evidence while AI work moves across models and tools. Its current product addresses two consequential boundaries. Prompt Lead establishes a designated, human-authored operating source before probabilistic execution. Pipeline Lab creates a deliberate inspection boundary before AI-assisted work is carried forward. Neither product claims to make a model deterministic, guarantee instruction adherence, or replace repository permissions, code review, testing, security controls, or deployment approval. Together, they make the operating context more explicit at the source and the continuation decision more visible at the handoff.

The category matters because the assumptions that governed conventional software delivery are changing. Requirements engineering historically separated the specification of work from its implementation. Source control preserved changes. Identity and access management defined who could act. CI/CD systems automated repeatable execution. Observability made system behavior visible. Security tooling inspected artifacts and dependencies. These controls remain essential, but AI introduces a probabilistic interpreter inside the delivery path. The system does not merely execute a deterministic specification. It interprets natural language, infers missing requirements, reconciles competing instructions, selects intermediate actions, and may revise its plan after observing tool output.

Research increasingly documents the consequences. Language models can infer omitted requirements, but those inferences can be fragile across prompt edits and model changes.[4] Meaning-preserving formatting differences can materially change model behavior.[6][7] Relevant information can be underused when it appears in the middle of long contexts.[8] Multi-turn interactions can produce premature assumptions from which models fail to recover.[9] Models can generate fluent code while missing explicit functional or non-functional constraints.[10][11] Instruction-hierarchy training and structured prompt practices improve these outcomes, but they do not turn a chat history into an authoritative organizational source or natural-language instructions into an enterprise permission system.[5][14][15]

This paper therefore separates two problems that are often collapsed.

- **Operating Source Integrity:** whether the designated human-authored source for a unit of AI-assisted work remains identifiable, reviewable, and reproducible before execution.
- **Execution Interpretation Risk:** whether a probabilistic model misunderstands, deprioritizes, conflicts with, or fails to follow part of that source.

Prompt Lead directly supports the first problem. It uses two explicit human-editable source fields - standing instructions and current context - and deterministically compiles their saved state without calling a model. The same supported saved source yields the same compiled brief. That does not make downstream interpretation deterministic, complete, correct, or compliant. It means the person and organization can identify the source they intended to use, inspect it, revise it, and deliberately carry it across tools without asking another model to silently rewrite it.

Pipeline Lab addresses the complementary boundary. It does not formally verify model alignment or authorize enterprise actions. It presents AI-assisted work for deliberate inspection through one visible provider path; it does not silently invoke a second provider or silently transfer output into Prompt Lead; and raw content is ephemeral by default. The person remains responsible for deciding what moves forward. Current Ludian V1 is deliberately narrower than universal runtime governance. It establishes governance **at the source and the handoff**.

That narrowness is useful now. Better models do not eliminate the need for governance. They increase the action surface, the number of consequential handoffs, the speed at which work can move, and the cost of ambiguous authority. The more a system can do, the more important it becomes to distinguish capability from permission.[16] The more work crosses tools, agents, and providers, the more important it becomes to preserve the context that should govern it. The more providers compete, specialize, reprice, and change, the more valuable it becomes for the operating source to outlive the model interpreting it.[1][26]

This is why AI software delivery needs its own governance layer. Governance can no longer remain only in policy documents, model settings, issue trackers, code scanners, or final approval steps. It must become infrastructure around the work: a durable operating source before execution, visible context during transition, evidence for inspection, and explicit human authority at consequential decisions.

Ludian's public promise and its governance thesis reinforce each other:

> **Confidence is the outcome. Governance is the mechanism.**

"The confidence layer for AI software delivery" describes the enterprise result: greater confidence that AI-assisted work remains connected to explicit human intent and deliberate review. "Governance for AI software delivery" describes the operational discipline required to produce that confidence across a changing AI stack.

Current V1 is a real product boundary, not a claim of completed enterprise governance. Prompt Lead and Pipeline Lab are Available. Decision Intelligence, the IP allowlist, and SCIM are Preview; SCIM is disabled in V1. SAML SSO, Custom RBAC, and Private Brain are Planned. The expansion path is from explicit operating context, to deliberate inspection, to organizational decision intelligence that can incorporate preferences, policy, evidence, risk, and human decision rights. The product must earn that future through use and evidence rather than imply it is already complete.

Abstract

AI-assisted software delivery increasingly spans models, agents, coding tools, review systems, repositories, deployment systems, and human teams. This expansion creates a governance problem that model-level safety, traditional requirements practices, post-hoc scanning, and ad hoc prompting do not fully address: the authoritative human intent, constraints, evidence, and decision authority surrounding the work can fragment as the work moves.

This paper argues that AI software delivery has become a distinct governance domain. It synthesizes research on software-engineering agents, prompt underspecification, prompt sensitivity, long-context behavior, multi-turn unreliability, instruction following, prompt injection, organizational governance, human oversight, provenance, and runtime governance. It distinguishes **Operating Source Integrity** - the integrity and reproducibility of the designated human-authored source before execution - from **Execution Interpretation Risk** - the risk that a model misreads or fails to follow that source.

The paper proposes a governance architecture spanning the lifecycle of AI-assisted work: governance before execution, governance during movement across systems, and governance after generation at inspection and continuation boundaries. It further proposes that governance will become infrastructure analogous to identity, source control, observability, and policy enforcement: an independent layer that remains useful as underlying models change.

Ludian V1 is examined as a focused implementation of governance at two boundaries. Prompt Lead deterministically compiles saved, human-authored operating sources without calling a model. Pipeline Lab creates a deliberate inspection and continuation boundary around AI-assisted work. These features do not eliminate ambiguity, downstream noncompliance, prompt injection, or the need for enterprise decision rights. They provide an external operating source and a visible handoff independent of any single model provider. The paper maps current support, future Decision Intelligence directions, limitations, and falsifiable research questions.

1. Software Delivery Has Changed

Software engineering is no longer merely assisted by AI at the level of autocomplete. SWE-bench formalized the ability of language models to address real-world GitHub issues across existing repositories, and human-in-the-loop systems such as Atlassian and Monash University's HULA have demonstrated workflows in which agents read Jira work items, create plans, write code, and raise pull requests under human guidance.[2][3] ReAct and related architectures explain why the change is structural: a model can reason, act through tools, observe results, and revise a plan across multiple steps.[27]

The result is a software-delivery system with a new probabilistic participant. The model does not merely run a deterministic command. It interprets a requirement, decides what information is salient, infers what is missing, proposes a plan, generates an artifact, and may choose its next action based on intermediate results. Even when the final commit is reviewed under familiar controls, important decisions have already occurred upstream.

This does not make conventional software governance obsolete. Requirements, architecture standards, source control, CODEOWNERS, branch protections, testing, security scanning, change approval, release management, and incident response remain essential. The issue is that they govern different objects. A repository can show what changed. A scanner can show whether a known vulnerability exists. A deployment control can show who approved release. Those systems may not show which human-authored operating brief governed the AI-assisted work, what assumptions entered during generation, which model or provider path was used, or what evidence the person considered before moving the work forward.

Stanford's 2026 AI Index describes a widening gap between capability and preparedness: technical performance and adoption are advancing while responsible-AI measurement, transparency, and governance implementation lag.[1] NIST's Generative AI Profile similarly frames governance as a lifecycle activity organized through Govern, Map, Measure, and Manage rather than as one final test.[22] The EU AI Act emphasizes risk management, record keeping, transparency, and human oversight for covered high-risk systems.[23] These frameworks are broader than software delivery, but their logic applies directly: governance must remain connected to context, lifecycle, risk, and accountable decisions.

The question is therefore not whether software teams already have governance. They do. The question is whether that governance remains coherent when a probabilistic model participates in the path from requirement to change.

A useful way to state the problem is:

> **AI software delivery is gaining capability faster than its governing context can travel.**

The governing context includes more than a prompt. It includes objectives, constraints, non-goals, architecture requirements, data boundaries, approval expectations, risk tolerance, evidence, and decision rights. As work moves through models, tools, agents, reviewers, and repositories, those elements can fragment even when each tool performs its local function correctly.

That fragmentation is the reason AI software delivery needs a governance layer of its own.

---

2. Requirements and Prompts Are Necessary - and Not Enough

The problem is not that requirements are obsolete or that prompts are useless. Both are essential. The problem is architectural: a requirement can exist without remaining attached to execution, and a prompt can guide execution without becoming a durable governance artifact.

2.1 Requirements were designed for a different execution assumption

Requirements engineering treats requirements as artifacts that should be clear, consistent, verifiable, and traceable.[12] That discipline remains relevant. But traditional requirements processes generally assume that a human team interprets the requirements and that implementation decisions become visible through familiar engineering artifacts: designs, tickets, commits, reviews, test results, and release approvals.

AI changes the interpretation layer. Natural-language requirements may be filtered through a model that can infer, summarize, prioritize, and transform them. A requirement stored in Jira or a specification can remain authoritative while the prompt actually supplied to the model contains only part of it. Conversely, a chat can accumulate instructions that were never formally approved as requirements. The system may therefore have several plausible sources of truth without a clear designated operating source for the AI-assisted task.

The governance problem is not solved by choosing “requirements” over “prompts.” It is solved by establishing a relationship among the authoritative requirements, the operating source actually supplied for the task, the probabilistic execution path, and the evidence reviewed before continuation.

2.2 A prompt is an execution input, not automatically a governance artifact

A prompt can be carefully written, saved, versioned, tested, and reused. Prompt engineering can improve model performance, and requirement-oriented training can help people articulate what they want more effectively.[5] Few-shot examples and instruction-hierarchy training can reduce some forms of sensitivity and conflict.[7][14] These are meaningful mitigations.

But prompt quality and governance are different properties.

A prompt is an instruction presented to a probabilistic interpreter. A governance artifact must also answer questions about authority and continuity: Which source is authoritative? Who changed it?

Which version governed the work? Which parts are enduring and which are task-specific? What moved to the next system? What evidence was inspected? Who decided to continue?

The literature documents why the distinction matters.

Underspecification. Yang and colleagues found that models can infer omitted requirements in tested settings, but the behavior is fragile: underspecified prompts were approximately twice as likely to regress across prompt or model changes, with some observed declines exceeding 20 percentage points. Simply specifying every requirement did not consistently help because requirements can conflict and models have limited instruction-following capacity.[4]

Human articulation. Ma and colleagues found that requirement-driven training improved how novice users articulated requirements for LLMs.[5] This supports an important counterpoint: better prompting helps. It also supports the need for an explicit requirements-oriented source rather than endless conversational optimization.

Prompt sensitivity. Sclar and colleagues showed that meaning-preserving differences in prompt formatting can produce large performance variation in tested settings.[6] POSIX provides a broader measure of prompt sensitivity and finds that model scale or instruction tuning alone does not necessarily eliminate it, while few-shot examples can reduce it.[7]

Long context. Lost in the Middle found that models can underuse relevant information depending on where it appears in a long context.[8] A larger context window does not by itself guarantee reliable use of every instruction.

Multi-turn unreliability. Laban and colleagues tested more than 200,000 simulated conversations and found an average decline of 39% across six generation tasks in the multi-turn condition. Their analysis attributed much of the loss to unreliability: premature assumptions and failure to recover after a wrong turn.[9]

Constraint following. FollowBench separates general response quality from satisfaction of individual content, situation, style, format, and example constraints.[10] CodeIF applies fine-grained instruction-following evaluation to code generation across function synthesis, debugging, refactoring, and code explanation.[11] These studies reinforce a practical truth: plausible output and compliant output are different evaluation problems.

Instruction hierarchy and injection. The Instruction Hierarchy paper argues that models can fail to distinguish privileged instructions from untrusted text; specialized training can improve robustness, but it does not turn natural-language instructions into enterprise authorization.[14] Greshake and colleagues demonstrated that instructions embedded in retrieved content can redirect LLM-integrated applications.[15]

Model changes. PromptBridge documents “model drifting”: a prompt optimized for one model can lose effectiveness when transferred to another, requiring adaptation.[26] This is evidence for model sensitivity, not a reason to claim that one prompt should work identically everywhere.

The synthesis is narrower and stronger than “prompts fail.”

> **Prompts are necessary execution inputs. They are not, by themselves, a durable governance substrate.**

2.3 Operating Source Integrity and Execution Interpretation Risk

This paper proposes two Ludian concepts.

Operating Source Integrity is the degree to which the designated human-authored source for a unit of AI-assisted work remains identifiable, reviewable, and reproducible before execution.

Execution Interpretation Risk is the risk that a probabilistic model misunderstands, deprioritizes, conflicts with, or fails to follow part of that source.

These concepts separate what can be stabilized from what remains probabilistic.

An organization can stabilize the source it intends to use. It can preserve the wording, separate enduring instructions from current context, record the version, and deliberately transfer the source. It cannot thereby guarantee how a downstream model will interpret the source. That interpretation remains a property of the model, prompt structure, context, tools, and execution path.

Prompt Lead supports Operating Source Integrity. Pipeline Lab creates a boundary for inspecting the result of Execution Interpretation Risk. Neither eliminates the risk.

Table 1. Requirements, prompts, and a governance layer

Dimension	Requirements or issue tracker	Conversational prompt	Governance layer for AI software delivery
Primary purpose	Define or coordinate work	Instruct a model	Preserve governing context around AI-assisted work
Authoritative source	Can be formal but may be distant from execution	May be distributed across turns	Designated operating source linked to the task
Transformation	Human/tool dependent	Probabilistic interpretation	Deterministic source preparation can precede probabilistic execution
Version clarity	Often supported	Often ambiguous in long chat histories	Should be explicit and reviewable
Cross-model portability	Requirements can be reused; prompt adaptation may be needed	Behavior can be model-sensitive	Preserve source while allowing model-specific adaptation
Permission	Defined elsewhere	Not reliably established by natural language	Must connect to external human and system authority
Evidence	Requirements and tickets provide partial context	Tool-local history may provide partial context	Should preserve evidence at consequential handoffs
Current Ludian V1	Complementary, not replaced	Downstream execution mechanism	Prompt Lead at source; Pipeline Lab at handoff

3. Governance Before, During, and After AI-Assisted Work

A governance layer becomes clearer when the software-delivery path is divided into three periods: before execution, during movement through AI systems, and after generation at consequential handoffs.

3.1 Before: define the operating source

Before AI-assisted work begins, the organization needs a designated source that answers:

- What is the objective?
- Which standing instructions apply?

- What is the current context?
- Which constraints are non-negotiable?
- What is outside scope?
- What evidence will be needed before continuation?
- Who is responsible for the decision to move forward?

Not every task requires a formal specification. The point is not bureaucracy. It is source clarity. If the task is consequential enough to govern, the operating source should not have to be reconstructed from memory after the model has acted.

Prompt Lead is Ludian's current implementation at this stage. It separates standing instructions from current context and compiles the saved source without a model call.

3.2 During: keep the path visible

Once a model or agent begins execution, new governance questions emerge:

- Which provider and model path is visible?
- Which tools and data can the system access?
- Did the work remain within scope?
- Did an instruction from a tool or retrieved source conflict with the operating source?
- Which intermediate decisions or changes matter?
- Does the risk of the task require escalation?

Runtime-governance research argues that path-dependent behavior cannot be fully governed through prompts or static access control alone.[17] Governance-by-design research shows that enterprise governance is implemented through concrete architectural and organizational arrangements: tool access, data access, memory design, update processes, accountability, and staged autonomy.[18]

Current Ludian V1 does not intercept every runtime action. That is an explicit boundary. Its provider path is visible, but full path governance and tool authorization remain future design directions.

3.3 After generation: inspect, decide, and preserve evidence

After a model produces work, the question is not merely whether the output exists. The organization needs a deliberate decision about what happens next.

- What was produced?
- Against which operating source should it be inspected?
- What tests, explanations, or evidence are available?
- What remains uncertain?
- Is the action reversible?
- Who is authorized to continue?

GAIE proposes graduated oversight based on regulatory impact, customer proximity, reversibility, and data sensitivity, with evidence requirements calibrated to risk.[19] The EU AI Act and NIST likewise emphasize human oversight, records, and lifecycle risk management in their respective scopes.[22][23]

Pipeline Lab is Ludian's current implementation at the handoff. It makes inspection visible and continuation deliberate. It does not provide formal verification, comprehensive provenance, or enterprise-wide authorization.

Table 2. Governance across the AI software-delivery lifecycle

Stage	Governance question	Current Ludian V1	Future direction
Before execution	What operating source should govern the work?	Prompt Lead	Organizational context, shared policy, Private Brain
During execution	Which path, tools, and constraints apply?	Visible provider path; no silent second provider	Runtime/path signals, provider constraints, risk classification
At inspection	What was produced and what should move forward?	Pipeline Lab	Decision evidence, comparative analysis, risk-sensitive review
At authorization	Who may approve or commit consequential work?	Human continuation is explicit	Custom RBAC, decision rights, escalation rules
After continuation	What evidence should remain?	Partial and task-local	Evidence continuity and broader audit context

The current architecture therefore governs two boundaries without pretending to govern the entire path. This is not the final category architecture. It is a practical starting point.

4. Governance Is Becoming Infrastructure

The most important strategic shift is not that governance will become another dashboard. It is that governance will become infrastructure around AI-assisted work.

Infrastructure performs a durable function independently of the application or model currently in use.

- Identity infrastructure establishes who or what is acting.
- Source control establishes what changed and preserves history.
- CI/CD infrastructure executes repeatable delivery processes.
- Observability infrastructure makes behavior visible.
- Security infrastructure constrains access and detects risk.
- Governance infrastructure connects intent, authority, evidence, and decision boundaries to the work.

These layers overlap but are not interchangeable. Logs can show what happened without showing why it was allowed. A repository can show the diff without showing the complete operating source. IAM can show that a person had access without showing that the person intended a specific AI-assisted change. A policy document can state a rule without remaining attached to the execution path.

A governance layer should therefore answer a distinct set of questions:

1. What human-authored operating source governed this unit of work?
2. Which provider, model, agent, and tool path was involved?
3. Which constraints and organizational context were relevant?
4. What evidence existed at the decision point?
5. Who decided that the work should continue?
6. What changed when the work moved to another system?

This framing explains why better models increase rather than eliminate the value of governance.

A more capable model can act across more of the lifecycle. It can interpret broader requirements, edit more files, invoke more tools, and produce work at a higher rate. That reduces the cost of generation while increasing the volume and consequence of decisions surrounding generation. The bottleneck moves from “Can the model produce something?” toward “Can the organization preserve context, inspect the work, and authorize the right continuation without recreating the entire path?”

Governance thus becomes an enabling layer. The goal is not to slow every action. Minimum viable governance research argues for controls proportionate to risk and embedded into operational work rather than imposed only as static barriers.[25] Governance by design makes a similar point at the architectural level: autonomy can scale only when accountability, tool access, memory, and update practices scale with it.[18]

This suggests a durable market logic.

Model capability will continue to change. Interfaces will change. Provider rankings will change. The need to know what was intended, what evidence existed, and who decided to continue will remain.

> **Models are replaceable components. The governing context around consequential work is an organizational asset.**

That does not mean a governance layer owns all context or becomes the only system of record. It means the layer must remain useful across model generations and tool boundaries.

---

5. What an AI Software Delivery Governance Layer Must Preserve

This paper proposes **Governance Continuity** as a Ludian synthesis: the ability to keep the governing context around AI-assisted work coherent as the work moves across systems.

A full governance layer requires at least five forms of continuity.

5.1 Intent Continuity

The objectives, constraints, requirements, non-goals, and operating brief remain identifiable across models, tools, agents, and teams.

Current Ludian support: Direct but bounded. Prompt Lead preserves and deterministically compiles the designated operating source.

Current limitation: The person can still omit, misunderstand, or contradict requirements. Ludian does not validate every element of intent.

5.2 Evidence Continuity

The information needed to inspect, understand, and justify what moves forward remains available at consequential handoffs.

Explicit-provenance research argues that responsibility becomes difficult to assign when composed systems do not produce lifecycle evidence.[20] CAVA explores canonical action identity and approval binding across heterogeneous runtimes.[21] These are future-facing frameworks, not descriptions of current Ludian V1.

Current Ludian support: Partial. Prompt Lead provides a stable brief; Pipeline Lab creates a deliberate inspection point.

Current limitation: Current V1 does not provide comprehensive cross-system provenance, durable evidence retention, or an enterprise evidence graph.

5.3 Authority Continuity

The workflow remains clear about who may approve, change, transfer, commit, or deploy consequential work.

Capability-versus-permission research formalizes the distinction between what an agent can do and the autonomy an organization chooses to allow based on risk, oversight, reversibility, and accountability. [16]

Current Ludian support: Partial. Continuation from Pipeline Lab is deliberate and human-controlled.

Current limitation: Current V1 does not implement full organizational decision-right mapping, Custom RBAC, or runtime permission enforcement.

5.4 Provider and Model Portability

A change in model or tool should not force the organization to reconstruct the operating source from scratch.

Current Ludian support: Partial. Prompt Lead can be deliberately carried across supported tools.

Current limitation: Prompt behavior remains model-dependent; adaptation may be necessary, and provider-specific features may not transfer.

5.5 Risk-Proportional Oversight

Inspection and human involvement should scale with consequence, reversibility, customer proximity, data sensitivity, and organizational risk.[19]

Current Ludian support: Not implemented as a dynamic automated control.

Future direction: Decision Intelligence can support risk-sensitive recommendations and escalation while keeping final authority human.

Table 3. Governance Continuity Framework

Framework element	Current V1	Preview / Planned direction	Not currently claimed
Intent Continuity	Prompt Lead source and deterministic compilation	Organizational context through Private Brain	Complete requirements correctness
Evidence Continuity	Brief plus deliberate inspection	Decision evidence and comparative analysis	Comprehensive provenance
Authority Continuity	Human decides what leaves Pipeline Lab	Custom RBAC and decision boundaries	Enterprise-wide authorization
Provider portability	Deliberate brief reuse across tools	Model/strategy comparison	Identical behavior across providers
Risk-proportional oversight	Manual inspection	Decision Intelligence and risk-sensitive recommendations	Autonomous runtime enforcement

The framework is deliberately larger than current V1. That distinction matters. A category paper should define the complete problem while being exact about the portion a current product implements.

6. Ludian Today: Governance at the Source and the Handoff

Ludian's mission is to make human intent, evidence, and authority durable across AI software delivery.

Its public promise is:

> **The confidence layer for AI software delivery.**

Its operational category is:

> **Governance for AI software delivery.**

The relationship is straightforward: confidence is the desired outcome; governance is the operating mechanism.

6.1 Prompt Lead: governance at the source

Prompt Lead has two explicit human-editable fields:

- standing instructions;
- current context.

The saved fields are compiled deterministically. Prompt Lead makes no AI-provider call during compilation. It does not ask a model to summarize, optimize, infer, or rewrite the source. The person can review the source and the compiled brief, save it deliberately, and carry it into another supported tool.

This creates current value in four ways.

A designated source. The user can identify which saved fields define the operating brief instead of reconstructing it from a conversation.

Reproducibility. The same supported saved source produces the same compiled artifact.

Reviewability. The source can be examined and corrected before model execution.

Portability. The compiled brief can be deliberately reused across tools, reducing the burden of recreating context from memory.

Prompt Lead does not guarantee that the brief is complete, unambiguous, organizationally authorized, or followed by the downstream model. Its value is the integrity and portability of the source artifact.

The distinction is essential:

> **Ludian does not make probabilistic models deterministic. It makes the human operating source explicit, reproducible, portable, and inspectable around them.**

6.2 Pipeline Lab: governance at the handoff

Pipeline Lab provides a deliberate inspection point for AI-assisted work. It uses one visible provider path. It does not silently call a second provider. It does not silently transfer output into Prompt Lead. Raw content is ephemeral by default.

This creates a visible handoff:

1. AI-assisted work is produced.
2. The person inspects the work and available context.
3. The person decides what, if anything, moves forward.

Pipeline Lab is not formal verification. It does not establish that the work satisfies every constraint, and it does not replace code review, testing, security scanning, repository controls, or deployment approval. It changes the default from invisible continuation to deliberate continuation.

6.3 The combination

Prompt Lead and Pipeline Lab matter together.

> **Prompt Lead preserves the operating source. Pipeline Lab makes continuation deliberate.**

That combination begins to create continuity around probabilistic execution without pretending to control the model from end to end.

The current product is therefore not accurately described as a prompt manager or a coding assistant. It does not generate the brief with a model, execute code autonomously, or compete with the model layer. It separates the human operating source from execution and pairs that source with an inspection boundary.

6.4 What Ludian does not replace

Ludian does not replace:

- requirements management;
- project or issue tracking;
- IDEs and coding assistants;
- source control;
- code review;
- application security testing;
- software supply-chain controls;
- IAM or repository permissions;
- CI/CD;
- deployment approval;
- enterprise GRC.

The governance layer connects context and decision boundaries across these systems. It is complementary infrastructure.

6.5 Why this is current value, not only future vision

The current value does not wait for a complete enterprise roadmap.

A person can preserve an explicit operating brief today. The same saved source can be compiled reproducibly. The brief can be deliberately reused across tools. AI-assisted work can be presented for inspection through a visible provider path. Continuation can remain a human decision rather than a silent transfer.

These are modest claims compared with universal runtime governance. They are also practical controls that exist now.

7. Why Better Models Make Governance More Valuable

It is tempting to assume that governance is a temporary response to immature models: as models improve, instruction failures decline, output quality rises, and governance becomes less necessary.

That conclusion confuses quality with authority.

A better model can follow instructions more reliably and produce higher-quality code. It can also operate across more files, use more tools, make more intermediate decisions, and create more changes per unit of human attention. Higher capability expands both the benefit and the consequence of delegation.

Three effects follow.

7.1 The cost of generation falls faster than the cost of judgment

AI can generate options, patches, explanations, tests, and refactors quickly. Human judgment remains scarce. The bottleneck shifts toward deciding which work reflects the operating brief, which evidence is sufficient, and which actions should move forward.

7.2 More capability creates more authority questions

Capability is not permission.[16] The ability to rewrite authentication logic, modify a schema, or propose a release does not grant the authority to commit or deploy it. Better models make this distinction more important because more actions become technically possible.

7.3 Provider competition increases portability value

Stanford's AI Index describes an active frontier with competing providers and changing capability, cost, and openness.[1] PromptBridge shows that prompts optimized for one model can lose effectiveness on another.[26] Enterprises may change providers because of capability, price, privacy, region, deployment model, procurement, or policy.

Source portability and behavioral portability are different. Ludian V1 contributes to source portability. It does not promise that the same brief produces the same code across models.

The defensible thesis is:

> **Model swappability becomes governance resilience when the authoritative operating source can outlive the model interpreting it.**

This can reduce reconstruction burden and make model-specific adaptation more disciplined. It does not eliminate switching cost.

8. The Enterprise and Strategic Case

The governance layer creates value at several levels without requiring the entire future platform to exist today.

8.1 Individual operator

- Maintains one explicit operating brief rather than reconstructing instructions repeatedly.

- Reuses that brief across tools.
- Inspects AI-assisted work before deliberately carrying it forward.
- Retains visible personal responsibility over handoffs.

8.2 Engineering and product leadership

- Gains a clearer operating source for AI-assisted work.
- Reduces dependence on provider-specific conversation history.
- Establishes a reviewable boundary between generation and continuation.
- Creates a foundation for more consistent team practices.

8.3 CTO, CIO, CISO, and CAIO

- Sees a provider-neutral architecture rather than another model dependency.
- Preserves explicit human involvement at two consequential boundaries.
- Gains a path toward risk-proportional oversight, organizational context, and decision evidence.
- Avoids representing current V1 as a compliance certification or autonomous enforcement system.

8.4 PE operating partner

- Can consider a common operating-brief discipline across portfolio companies without mandating one model provider.
- Can evaluate model switching from a governance perspective, not only procurement.
- Gains a potential path toward comparable AI-delivery decision evidence across businesses.

8.5 Investor and strategic platform

- Sees a focused product wedge with current utility.
- Sees expansion toward Decision Intelligence and organizational governance.
- Sees strategic adjacency to cloud, developer tools, SDLC, work management, security, observability, governance, and enterprise services.
- Does not need to assume Ludian is publicly seeking an acquirer.

The commercial logic is a progression from individual operating discipline, to team inspection and shared context, to enterprise decision intelligence and controls. That progression must be proven through adoption, retention, enterprise evaluations, and customer evidence. It is not a current traction claim.

8.6 Category landscape

The language of governed AI-assisted software delivery is beginning to appear across products and services. Hakama describes a governance layer around AI-assisted delivery; code4thought uses “AI Software Delivery Governance”; ZeroUI describes a software-engineering governance operating system. [28][29][30] Ludian does not claim first use of the category.

Its specific position is different and narrower in V1: preserve the human operating source without a model call, make that source deliberately portable, and pair it with a deliberate inspection boundary. That is the category contribution this paper advances.

9. From Current V1 to Decision Intelligence

The current product and roadmap must remain clearly separated.

Available

- Prompt Lead
- Pipeline Lab

Preview

- Decision Intelligence
- IP allowlist
- SCIM, disabled in V1

Planned

- SAML SSO
- Custom RBAC
- Private Brain

The future architecture is not “more automation for its own sake.” It is a broader governance layer capable of incorporating:

- organizational preferences and context;
- model and strategy comparison;
- evidence-aware recommendations;
- risk-sensitive review and escalation;
- decision-right awareness;
- provider and deployment constraints;
- human-controlled acceptance and override.

Decision Intelligence should recommend and explain; people remain responsible for consequential decisions. Private Brain can provide tenant-owned organizational context. Custom RBAC can express roles and decision boundaries. SAML and SCIM can support enterprise identity. These are directions, not retroactive claims about V1.

A mature governance layer may eventually connect all three time horizons:

- **Before AI:** preserve the operating source and authority context.
- **During AI:** observe paths, tools, constraints, and risk.
- **After AI:** preserve decision evidence and route consequential continuation.

Current Ludian V1 establishes the source and handoff architecture on which that broader layer can be built.

---

10. Limitations and a Falsifiable Research Agenda

A governance paper should be explicit about what could prove its thesis incomplete or wrong.

10.1 Current limitations

- Prompt Lead cannot ensure the user has written a complete or correct brief.

- Deterministic compilation does not make model interpretation deterministic.
- A downstream model may ignore, misread, or conflict with stated instructions.
- Pipeline Lab depends on human attention and judgment.
- Pipeline Lab is not formal verification.
- Current evidence continuity is partial.
- Current authority continuity is personal and procedural, not enterprise-wide.
- Manual portability can create friction.
- Provider-specific functions may not translate cleanly.
- Human oversight can become bureaucracy if poorly designed.
- Current product value has not yet been established through large-scale published empirical results.

10.2 Proposed studies

Operating-brief reconstruction study

Compare users reconstructing instructions from multi-turn history with users reusing a saved Prompt Lead. Measure omitted constraints, wording variance, time, conflict detection, and confidence.

Single compiled brief versus multi-turn disclosure

Present the same requirements either as one compiled brief or across multiple turns. Measure adherence, recovery from clarification, and user effort across current frontier model families.

Cross-model brief transfer

Use the same operating source across at least three current frontier model families and one enterprise-approved or open-weight model. Separate source consistency from output variance and model-specific adaptation.

Inspection-boundary study

Compare direct continuation from an AI assistant with deliberate Pipeline Lab inspection. Measure defects noticed, constraint violations noticed, false positives, time cost, and changed decisions.

Governance Continuity metric

Develop and test a metric spanning intent retained, evidence available, authority explicit, provider-transition effort, and inspection coverage. A useful metric must avoid rewarding documentation volume over decision quality.

A result showing no meaningful improvement in source reconstruction, inspection quality, or provider-transition burden would weaken the product thesis. Ludian should seek that evidence rather than assume it.

Conclusion

AI software delivery is becoming a distinct operating system inside the enterprise. It connects human requirements, probabilistic models, autonomous or semi-autonomous agents, developer tools, repositories, review systems, and delivery controls. Every component can become more capable while the governing context among them becomes less coherent.

The research is clear on the underlying difficulty. Prompts can be effective, but they can also be underspecified, format-sensitive, position-sensitive, model-sensitive, vulnerable to conflicting

instructions, and unreliable across long conversations. Models can produce useful code while missing stated constraints. Model-level mitigations and better prompting improve these problems; they do not make a chat history an authoritative organizational source or turn natural language into an enterprise permission system.

Conventional software controls remain indispensable. Requirements describe work. Repositories preserve changes. Scanners inspect code. IAM constrains access. CI/CD executes delivery. Governance for AI software delivery connects a different set of questions across those systems: What human-authored source governed the work? Which path was involved? What evidence existed? Who decided it could continue? What must remain coherent if the model changes?

Ludian's current answer is deliberately concrete. Prompt Lead separates a designated human-authored operating source from probabilistic execution and compiles that source without a model call. Pipeline Lab creates a visible point of inspection before continuation. Neither feature eliminates model risk. Together, they make governance more explicit at the source and the handoff.

That architecture creates current value: less reconstruction of operating context, more deliberate transfer across tools, visible human involvement, and a provider-neutral foundation for future decision intelligence. It also creates a disciplined path forward. Evidence continuity, enterprise decision rights, risk-proportional oversight, and runtime governance can be added without pretending they already exist.

The case for a governance layer becomes stronger as models improve. More capable systems can take more consequential actions, move more work through the pipeline, and consume more of the organization's scarce review capacity. Better models change the execution layer. They do not remove the need for an authoritative operating source, evidence at the decision point, or explicit human authority.

AI capability can travel on its own. Governance must be designed to travel with it.

Product Boundary

Status	Capability
Available	Prompt Lead
Available	Pipeline Lab
Preview	Decision Intelligence
Preview	IP allowlist
Preview, disabled in V1	SCIM
Planned	SAML SSO
Planned	Custom RBAC
Planned	Private Brain

References

[1] Stanford Institute for Human-Centered Artificial Intelligence. “The 2026 AI Index Report,” including Technical Performance, Responsible AI, Economy, and Policy and Governance chapters. Stanford University, 2026. https://hai.stanford.edu/ai-index/2026-ai-index-report

[2] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. “SWE-bench: Can Language Models Resolve Real-World GitHub Issues?” ICLR 2024. https://openreview.net/forum?id=VTF8yNQM66

[3] Wannita Takerngsaksiri, Jirat Pasuksmit, Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Ruixiong Zhang, Fan Jiang, Jing Li, Evan Cook, Kun Chen, and Ming Wu. “Human-In-the-Loop Software Development Agents.” ICSE-SEIP 2025, 342-352. DOI: 10.1109/ICSE-SEIP66354.2025.00036. https://arxiv.org/abs/2411.12924

[4] Chenyang Yang, Yike Shi, Qianou Ma, Michael Xieyang Liu, Christian Kaestner, and Tongshuang Wu. “What Prompts Don’t Say: Understanding and Managing Underspecification in LLM Prompts.” Findings of ACL 2026, 9072-9101. DOI: 10.18653/v1/2026.findings-acl.441. https://aclanthology.org/2026.findings-acl.441/

[5] Qianou Ma, Weirui Peng, Chenyang Yang, Hua Shen, Kenneth Koedinger, and Tongshuang Wu. “What Should We Engineer in Prompts? Training Humans in Requirement-Driven LLM Use.” ACM Transactions on Computer-Human Interaction 32, no. 4 (2025): 1-27. DOI: 10.1145/3731756. https://dl.acm.org/doi/10.1145/3731756

[6] Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. “Quantifying Language Models’ Sensitivity to Spurious Features in Prompt Design or: How I Learned to Start Worrying about Prompt Formatting.” ICLR 2024. https://openreview.net/forum?id=RIu5lyNXjT

[7] Anwoy Chatterjee, H. S. V. N. S. Kowndinya Renduchintala, Sumit Bhatia, and Tanmoy Chakraborty. “POSIX: A Prompt Sensitivity Index for Large Language Models.” Findings of EMNLP 2024, 14550-14565. DOI: 10.18653/v1/2024.findings-emnlp.852. https://aclanthology.org/2024.findings-emnlp.852/

[8] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. “Lost in the Middle: How Language Models Use Long Contexts.” Transactions of the Association for Computational Linguistics 12 (2024): 157-173. DOI: 10.1162/tacl_a_00638. https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00638/119934/Lost-in-the-Middle-How-Language-Models-Use-Long

[9] Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. “LLMs Get Lost in Multi-Turn Conversation.” arXiv preprint, 2025. https://arxiv.org/abs/2505.06120

[10] Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. “FollowBench: A Multi-level Fine-grained Constraints Following Benchmark for Large Language Models.” ACL 2024, 4667-4688. DOI: 10.18653/v1/2024.acl-long.257. https://aclanthology.org/2024.acl-long.257/

[11] Kaiwen Yan, Hongcheng Guo, Xuanqing Shi, Shaosheng Cao, Donglin Di, and Zhoujun Li. “CodeIF: Benchmarking the Instruction-Following Capabilities of Large Language Models in Code Generation.” ACL 2025 Industry Track, 1272-1286. DOI: 10.18653/v1/2025.acl-industry.89. https://aclanthology.org/2025.acl-industry.89/

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

[13] Zhenpeng Chen, Chong Wang, Weisong Sun, Xuanzhe Liu, Jie M. Zhang, and Yang Liu. “Promptware Engineering: Software Engineering for Prompt-Enabled Systems.” arXiv preprint, 2025. https://arxiv.org/abs/2503.02400

[14] Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. “The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions.” arXiv preprint, 2024. https://arxiv.org/abs/2404.13208

[15] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. “Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection.” ACM CCS 2023, 79-94. DOI: 10.1145/3576915.3623143. https://dl.acm.org/doi/10.1145/3576915.3623143

[16] Haining Zheng, Qian Dong, Rodolfo K. Depena, Jonathan D. Bhatia, Feng Xiao, and Peng Xu. “Separating Capability from Permission: A Governance Framework for Agentic AI Autonomy Levels.” arXiv preprint, 2026. https://arxiv.org/abs/2607.23438

[17] Maurits Kaptein, Vassilis-Javed Khan, and Andriy Podstavnychy. “Runtime Governance for AI Agents: Policies on Paths.” arXiv preprint, 2026. https://arxiv.org/abs/2603.16586

[18] Nelly Dux, Cristina Alaimo, Philippe Roussiere, and Abhishek Kumar Mishra. “Governance by Design: Architecting Agentic AI for Organizational Learning and Scalable Autonomy.” arXiv preprint, 2026. https://arxiv.org/abs/2605.20210

[19] Richard Kang. “Governed AI-Assisted Engineering: Graduated Human Oversight for Agentic Code Generation in Regulated Domains.” arXiv preprint, 2026. https://arxiv.org/abs/2606.22484

[20] Jinwei Hu, Xinmiao Huang, Qisong He, Youcheng Sun, Yi Dong, and Xiaowei Huang. “Responsible Agentic AI Requires Explicit Provenance.” arXiv preprint, 2026. https://arxiv.org/abs/2605.17169

[21] Zexun Wang. “CAVA: Canonical Action Verification and Attestation for Runtime Governance of Agentic AI Systems.” arXiv working paper, 2026. https://arxiv.org/abs/2607.13716

[22] Chloe Autio, Reva Schwartz, Jesse Dunietz, Shomik Jain, Martin Stanley, Elham Tabassi, Patrick Hall, and Kamie Roberts. “Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile.” NIST AI 600-1, July 26, 2024. DOI: 10.6028/NIST.AI.600-1. https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence

[23] European Parliament and Council. “Regulation (EU) 2024/1689 Laying Down Harmonised Rules on Artificial Intelligence.” Official Journal of the European Union, 2024. https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng

[24] OWASP GenAI Security Project. “Securing Agentic Applications Guide 1.0” and “OWASP Top 10 for Agentic Applications 2026.” 2025-2026. https://genai.owasp.org/resource/securing-agentic-applications-guide-1-0/

[25] Nick van der Meulen, Jennifer Jewer, and Nadège Levallet. “Minimum Viable Governance for Generative AI.” MIT Center for Information Systems Research, March 19, 2026. https://cisr.mit.edu/publication/2026_0301_GenAIGovernance_VanderMeulenJewerLevallet

[26] Yaxuan Wang, Quan Liu, Zhenting Wang, Zichao Li, Wei Wei, Yang Liu, and Yujia Bao. “PromptBridge: Cross-Model Prompt Transfer for Large Language Models.” arXiv preprint, 2025; accepted to COLM 2026. https://arxiv.org/abs/2512.01420

[27] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. “ReAct: Synergizing Reasoning and Acting in Language Models.” ICLR 2023. https://arxiv.org/abs/2210.03629

[28] Hakama. “Governed AI Software Delivery.” Accessed August 10, 2026. https://hakama.ai/

[29] code4thought. “AI Software Delivery Governance.” Accessed August 10, 2026. https://code4thought.eu/solutions-ai/ai-software-delivery-governance/

[30] ZeroUI. “Prevent-First Software Engineering Governance Operating System.” Accessed August 10, 2026. https://www.zeroui.dev/