Janus: A Minimal Governance Kernel for Deterministically Reconstructible Human–AI Development Systems

Lic. Martín Nicolás Sánchez Morales

Founder, Janus Governance Project
Argentina
https://janusgovernance.org

***Abstract*—The rapid adoption of artificial-intelligence-assisted development tools has significantly transformed software engineering workflows. However, current technology governance frameworks primarily operate at organizational or regulatory levels—focusing on compliance, ethics, and risk management—while leaving the day-to-day governance of interactions between human developers and generative code systems largely undefined.**

This paper introduces Janus, a minimal governance kernel for AI-assisted development environments. Janus defines a small set of foundational governance invariants that allow hybrid human–AI workflows to become traceable, auditable, and deterministically reconstructible.

The model is based on three foundational elements: append-only logs that preserve the institutional history of the system, an explicit dual evidence model (E^+ / E^-) capable of representing both recorded actions and verifiable omissions, and a formal boundary of human authority that maintains accountability for decisions requiring responsibility.

Unlike development methodologies such as Agile, DevOps, or ITIL, Janus does not redefine organizational processes. Instead, it introduces minimal operational invariants that stabilize the interaction between humans and generative systems without significantly increasing system complexity.

The result is a micro-structural intervention that can integrate with different technological stacks and reduce silent drift, ambiguity of authority, and loss of traceability in AI-assisted software development systems.

***Index Terms*—AI governance, human–AI systems, software governance, deterministic reconstruction, sociotechnical systems, development governance**

I. INTRODUCTION

Modern AI-based code generation tools—including conversational assistants integrated into development environments, automated copilots, and low-code automation platforms—have introduced a new production paradigm in software engineering.

This paradigm is characterized by:

- external generation of technical artifacts
- accelerated iteration cycles
- reduced syntactic friction
- partial reliance on automated reasoning

However, this transformation introduces a structural problem that remains largely unexplored:

How is the human–AI interaction governed at the micro-operational level of software development?

Traditional methodologies regulate organizational processes, deployment pipelines, and workflow coordination, but they do not explicitly formalize the relationship between automated code generation and human validation.

This gap introduces structural risks such as:

- diffuse authority
- undetected version drift
- validation based on plausibility
- absence of verifiable structural evidence

Janus emerges as a response to this governance gap.

A. Contributions

This paper makes three primary contributions:

1) It proposes a **minimal governance kernel** for AI-assisted development environments.
2) It introduces a **dual evidence model (E^+ / E^-)** capable of representing both recorded actions and structurally detectable omissions.
3) It defines a **deterministic reconstruction model** that enables governance states to be rebuilt from governed events.

These contributions aim to formalize micro-operational governance in hybrid human–AI systems without introducing additional organizational complexity.

B. Research Hypothesis

This work is grounded in the following hypothesis:

In AI-assisted development systems, the absence of verifiable structural evidence generates operational drift, incoherence, and ambiguity of responsibility. Introducing a minimal set of governance invariants—based on append-only records, explicit evidence models (E^+ / E^-), and auditable human decisions—transforms plausibility-based workflows into deterministically reconstructible systems.

In other words, the critical distinction is not merely between automation and human oversight, but between **believing that something occurred within a system and being able to demonstrate, through structural evidence, what actually occurred**.

Janus proposes a micro-structural intervention aimed at reducing this epistemic gap within hybrid human–AI systems. The underlying premise is that **small structural interventions can produce disproportionate improvements in the stability of complex systems**, particularly when human and machine actors share operational responsibility.

II. THEORETICAL BACKGROUND

A. Sociotechnical Systems

Sociotechnical systems theory states that the introduction of new technologies alters work organization and requires the redefinition of coordination and authority structures [1].

In AI-assisted development environments, the tool does not replace the developer; instead, it introduces an intermediate generative agent whose role and authority must be structurally defined. When generative systems participate directly in technical production, governance mechanisms must explicitly define how authority, validation, and responsibility are exercised across the human–AI boundary.

Within this context, the deterministic reconstruction model proposed by Janus also enables the preservation of **institutional memory** in hybrid socio-technical systems. Because governance events, evidence records, and human authority decisions are stored in append-only structures, the operational history of the system can be reconstructed over time without relying on informal recollection or fragmented documentation. In this sense, governance logs function not only as audit artifacts but as a durable record of institutional knowledge and decision history.

B. Governance and Responsibility

Governance involves the explicit definition of rules, responsibilities, and validation mechanisms within complex systems [2]. In software engineering literature, governance has also been examined at the architectural level, where architectural governance defines how system evolution and design decisions are supervised and validated across an organization [5].

In hybrid human–AI environments, the absence of clear operational rules may lead to reliance on informal trust or implicit authority.

Janus does not propose a general ethical framework but rather an operational model of technical governance.

C. Infrastructure-Independent Architecture

Clean architecture principles propose separating system rules from implementation technologies [3].

Janus adopts this approach by separating three components:

- Normative core
- Executable runtime
- Interaction protocols

This architecture allows governance rules to remain independent of any specific technology.

III. GOVERNANCE GAP IN AI-ASSISTED DEVELOPMENT

In AI-assisted development workflows it is common to observe the following cycle:

1) Requesting code from an AI system
2) Inserting generated code into the development environment
3) Iterating through trial-and-error
4) Declaring success based on observable functionality

This model may produce:

- silent structural drift
- version fragmentation
- loss of traceability
- undocumented implicit validation

Artificial intelligence does not create these problems by itself; rather, it amplifies pre-existing structural weaknesses.

IV. JANUS: MICRO-OPERATIONAL GOVERNANCE

Janus proposes a minimal governance layer based on the following principles:

- Explicit authority
- Baseline freezing
- Validation gates
- Structured telemetry
- Structural identity validation
- Deterministic reconstruction

Janus does not impose an organizational methodology. Instead, it defines **minimal structural invariants**.

A. Governance Kernel

Janus can be understood as a **minimal governance kernel** for hybrid human–AI systems.

This kernel defines only the foundational governance invariants necessary for maintaining traceability, auditability, and explicit responsibility.

1) Append-Only Logs: Three canonical logs store the system's institutional history:

- MANAGEMENT_LOG
- SCHEMA_LOG
- AUDIT_LOG

2) Evidence Model: The system distinguishes between two types of evidence:

- E^+ — explicitly recorded evidence
- E^- — evidence derived from the verifiable absence of an expected record

3) Governance Evaluation: Evaluation processes interpret evidence within a defined scope to determine whether governance events must be generated.

4) Governance Events: Janus defines two canonical events:

- OMISSION_DETECTED
- HUMAN_DECISION

5) Authority Boundary: Governance events may only be written through an authorized mechanism known as the **Audit Writer**.

6) Deterministic Reconstruction: Governance state can be reconstructed from:

- historical logs
- schema context
- evidence references

V. EVIDENCE MODEL (E^+ / E^-)

A central contribution of Janus is the introduction of a dual evidence model.

A. Positive Evidence (E^+)

Records actions that effectively occurred.

B. Negative Evidence (E^-)

Records detectable omissions: expected actions that did not occur within a defined structure.

This model allows representing not only events that occurred but also **structurally relevant absences**.

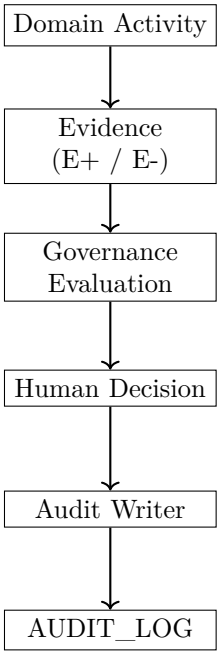

Fig. 1. Janus Governance Kernel Pipeline. Domain activity generates evidence (E+ / E), which is interpreted by governance evaluation flows. When accountability is required, explicit human authority produces a HUMAN_DECISION event. Governance outcomes are recorded through the authorized Audit Writer into AUDIT_LOG, enabling deterministic reconstruction of governance state.

C. Conceptual Formalization

$$\text{GovernanceState} = f(\text{Event}, \text{Evidence}, \text{HumanDecision}) \tag{1}$$

Where:

- **Event** represents domain events occurring in the system
- **Evidence** represents evidence generated under the E^+ / E^- model
- **HumanDecision** represents explicit human authority decisions

A second relation captures deterministic reconstruction:

$$\text{GovernanceState}_t = g(\text{MANAGEMENT\_LOG}_{\leq t}, \text{SCHEMA\_LOG}_{\leq t}, \text{AUDIT\_LOG}_{\leq t}) \tag{2}$$

This equation expresses that the governance state at time t can be deterministically reconstructed from the canonical logs of the system.

Evidence itself can be defined as:

$$\text{Evidence} = E^+ \cup E^- \tag{3}$$

Together, these relations show that governance in Janus does not rely on implicit states but on reconstructible functions derived from observable records.

$$\text{SystemState} = h(\text{GovernanceState}) \tag{4}$$

This relation expresses that, when governance events fully describe authorized system activity, the operational state of the system becomes derivable from the governance state itself. In other words, deterministic reconstruction assumes that governance events capture the set of authorized operational transitions within the system.

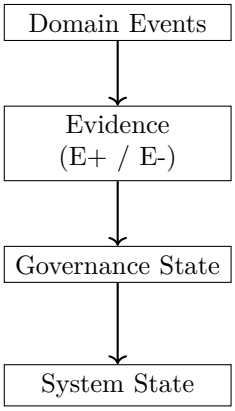

Fig. 2. Socio-Technical System Reconstruction Model implied by the Janus governance kernel. Domain events generate evidence (E+ / E), which is evaluated through governance processes. Human authority decisions and audit-controlled logging produce a governance state from which the operational system state becomes derivable.

VI. INITIAL EMPIRICAL VALIDATION

A relevant methodological characteristic of the project is that the development of Janus itself—including the conceptual core, code repositories, documentation environment, and experimental runtimes—was conducted under the same governance protocols proposed by the system.

In this sense, the project functioned as an initial case of **structural self-governance (dogfooding)**, where traceability rules, event recording, and human decisions were applied during the system's own construction.

This approach allowed the internal coherence of the model to be evaluated in real AI-assisted development conditions.

In methodological terms, **Janus was developed under its own governance constraints**, reinforcing the alignment between the proposed model and its experimental implementation.

The model was tested through:

- definition of a runtime-independent core
- implementation in an Apps Script runtime
- implementation in a Node runtime
- structured telemetry testing focused on verifying deterministic reconstruction of governance state from append-only logs and detecting structural inconsistencies through the E^+ / E^- evidence model

Observed results include:

- structural stability
- contract coherence
- structural detection of inconsistencies
- controlled reproducibility

VII. DISCUSSION

Although Janus is presented primarily as a **minimal governance kernel**, its design allows higher layers to evolve toward a broader **governance framework for AI-assisted development environments**.

This paper does not claim the existence of such a framework but instead describes the **minimal foundational governance invariants** that could support future extensions. In alignment with this objective, the Janus core is intended to be distributed as an **open-source governance kernel**, allowing independent researchers and engineering teams to reproduce experiments, audit the governance model, and implement alternative runtimes while preserving the canonical governance invariants defined in this work.

In this sense, Janus may be understood as a conceptual infrastructure upon which policies, tools, and governance runtimes could be constructed while preserving the simplicity of the proposed kernel.

Janus does not compete with organizational governance frameworks nor attempt to replace agile methodologies or architectural governance models described in the literature [5].

Instead, it operates at a different level: the **micro-operational layer of human–AI interaction**. At this level, deterministic reconstruction also implies that governance records function as a persistent institutional memory of the system, preserving the traceable history of decisions, evidence, and authority.

A. Conceptual Comparison

A central aspect of Janus is how it differs from existing models.

Most frameworks used in software engineering operate at different levels of the problem addressed in this work.

Methodologies such as **Agile, DevOps, and ITIL** regulate organizational processes and delivery cycles. Systems such as **Git** provide historical integrity of source code. Contemporary **AI governance frameworks** often focus on ethical, regulatory, or institutional dimensions, such as the NIST AI Risk Management Framework [4].

Janus instead operates at the **micro-operational level of human–AI interaction during technical production**.

Its distinctive characteristics include:

1) **Evidence-based governance** through the E^+ / E^- model
2) **Deterministic reconstruction of governance state** via append-only canonical logs
3) **Explicit separation between technical generation and human authority** through auditable decision events

TABLE I
CONCEPTUAL COMPARISON

Concept	Versioning	Organizational Governance	Janus
System History	commits	process documentation	append-only logs
Integrity	hashes	institutional audit	controlled audit writing
Validation	pull requests	hierarchical approval	HUMAN_ DECISION
Evidence	commit history	reports	E+ / E-
Reconstruction	checkout	documentary reconstruction	deterministic reconstruction
Operational Level	code organization	human governance	human–AI interaction

VIII. LIMITATIONS

Janus:

- does not guarantee system security
- does not replace regulatory compliance
- does not eliminate human error
- does not substitute organizational ethical frameworks

Its scope is deliberately limited.

IX. CONCLUSION

The integration of artificial intelligence into software development introduces new ambiguities regarding authority, responsibility, and validation.

While existing governance frameworks operate primarily at institutional levels, the everyday interaction between human developers and generative systems remains largely under-formalized.

Janus proposes a micro-governance layer designed to address this operational gap. Rather than introducing complex infrastructures, Janus defines a small set of foundational governance invariants capable of stabilizing hybrid human–AI systems.

The central hypothesis is that **small structural interventions can produce disproportionate improvements in the stability of complex systems**.

Similar to how distributed version control, code review practices, and continuous integration reshaped software engineering culture, a minimal governance layer may help reduce silent drift, ambiguity of responsibility, and traceability loss in AI-assisted development.

Janus therefore aims not to replace existing governance frameworks but to complement them by addressing the micro-operational layer where human and machine collaboration actually occurs.

If governance events fully describe authorized system activity, the operational state of the system becomes derivable from the governance state. In this sense, governance records are not merely audit artifacts but may function as a reconstructible structural description of system operation.

This interpretation highlights a broader implication of the model: governance infrastructure can also operate as a durable mechanism for preserving institutional memory in socio-technical systems. By recording governance events, evidence structures, and explicit human decisions, the system preserves a reconstructible history of institutional knowledge and operational responsibility.

References

[1] E. L. Trist and K. W. Bamforth, "Some Social and Psychological Consequences of the Longwall Method," *Human Relations*, 1951.

[2] J. Kooiman, *Governing as Governance*. Sage Publications, 2003.

[3] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall, 2017.

[4] National Institute of Standards and Technology, "AI Risk Management Framework (AI RMF 1.0)," NIST, 2023.

[5] G. Booch, "Architectural Governance in Software Systems," *IEEE Software*, 2006.