AI Governance Is Not a Policy. It Is an Architecture.

Compliance frameworks are necessary. They are not sufficient to make a regulated system structurally governable.

Twingital Institute / Jérôme Vetillard /April 2026

Introduction

A paradox has been running through the executive committees of the healthcare and life sciences sector for several years now. On one side, a large majority of organizations claim to have initiated structured AI governance programs. On the other, an equally significant proportion expresses a persistent sense of inadequacy when confronted with the concrete requirements of oversight, traceability, audit, and accountability attached to the systems they deploy. Recent market and ecosystem surveys document this gap recurrently: the DiMe Society reported in early 2026 that a large majority of executives surveyed still considered the insufficiency of guidelines a significant barrier to AI adoption, while the 2026 Larridin report simultaneously flagged low levels of complete AI application inventories, the frequent absence of a clear operational governance framework, and high levels of declared confidence in AI impact. These surveys document a structural malaise.

This paradox is not merely circumstantial. It reveals a confusion that is rarely articulated with sufficient clarity: the confusion between a governance «policy» and a governance «architecture».

These two objects do not belong to the same register. They do not represent two degrees of maturity on a single continuum. They are two distinct programs, addressing distinct problems, with distinct instruments. A governance policy organizes responsibilities, rules, procedures, evaluations, and escalation mechanisms. It frames the system from the outside. A governance architecture, by contrast, defines the structural properties that make certain behaviors traceable, bounded, or structurally unacceptable by design. It conditions the system from the inside.

The thesis of this article is as follows: in AI systems operating in regulated environments with high individual decision stakes, procedural governance frameworks are necessary but structurally insufficient. They enable documentation, evaluation, monitoring, and risk management. They do not guarantee, on their own, that a deployed system is traceable by construction, bounded within its validity domain, or that the boundary between automation and human decision-making will hold under real operational pressure. When these properties are not built into the design, governance remains retrospective, partial, and vulnerable to the workarounds that any organization under stress will mechanically produce.

This thesis has an explicit domain of validity. It applies to AI systems deployed in contexts where an individual decision can carry significant clinical, regulatory, economic, or organizational consequences: software medical devices, diagnostic or prognostic decision support, regulatory toxicological prediction, triage, healthcare resource allocation. It does not apply with the same intensity to low-stakes systems, for which a primarily procedural governance may, in some cases, suffice.

I. Terminological Clarification

The first requirement is to stabilize the terms, since the problem begins precisely with their confusion.

I use «procedural governance» to refer to the set of rules, committees, reference frameworks, evaluations, checklists, and documentary processes applied to an AI system by actors who observe it, evaluate it, and frame its use from outside its structure. It is indispensable. It intervenes primarily as a control and oversight mechanism, after the system's architecture has been established.

I use «architectural governance» to refer to the set of structural properties of a system that make its behavior traceable, its validity perimeter explicable, and its modes of action controllable at the level of the system's own functioning. It is not added to the system after design. It is embedded in its composition patterns, data flows, invariants, execution conditions, and recording mechanisms.

Two further distinctions are necessary.

The first opposes «reported auditability» and «native auditability». By reported auditability, I mean the capacity to reconstruct a decision after the fact from artifacts produced around the system: application logs, metadata, observability tools, local explainability approximations. By native auditability, I mean the property of a system whose ordinary functioning itself produces the elements necessary for faithful reconstruction of the execution context: data engaged, relevant state transitions, component versions, decision regime applied.

A nuance must be introduced here that is insufficiently present in the literature on this subject. Native auditability does not mean complete explainability of the internal mechanism of a statistical model. That would be an untenable claim for any model of non-trivial complexity. It means constitutive traceability of the context, inputs, outputs, transitions, and execution regime of the decision. It decisively improves the governability of a system. It does not, on its own, eliminate the difficulties of intelligibility of the inferential mechanisms themselves. This point deserves to be stated at the outset, to avoid conflating two distinct problems that the industry tends to superpose when it wishes to sidestep one by feigning to resolve the other.

The second distinction is this: a system that can be inspected is not yet a system that renders itself structurally inspectable. Current frameworks primarily reinforce inspection capabilities. The question posed here is different, and more demanding: how do we design systems whose normal operation makes that inspection reliable, continuous, and technically grounded?

II. Why Procedural Governance Dominates and Why That Is No Longer Sufficient

The current prevalence of procedural governance is neither accidental nor absurd. It is the coherent product of an industrial and regulatory history.

The major AI governance frameworks (the NIST AI Risk Management Framework, the ISO/IEC 42001 standard, and the graduated obligations introduced by the European AI Act) were all built according to a widely shared principle of technological neutrality. This choice is understandable from a regulatory standpoint: locking a specific architectural pattern into a legal or normative text would risk freezing innovation within a dated design model. It is therefore logical that these texts impose functional, documentary, and organizational obligations without prescribing a particular system design. The NIST AI RMF, for example, is explicitly structured as a risk management framework around the functions Govern, Map, Measure, and Manage. The EU AI Act, meanwhile, provides for a phased application schedule

depending on obligation categories, with general application on August 2, 2026, and a longer transition for certain categories, notably high-risk systems integrated into regulated products.

This choice has had the effect, in many organizations, of reinforcing a reading whereby governance could be added after design, through procedural and documentary superimposition, on systems built primarily for functional performance or delivery speed. Since the text does not prescribe how to build, it became tacitly permissible to build first, then add governance in successive layers. This is where the structural bias forms.

To this regulatory bias, a cultural bias must be added. Much of contemporary software engineering has been shaped by logics of rapid iteration and incremental correction. This culture is highly effective in contexts where errors are tolerable, reversible, and dissociated from significant individual stakes. It becomes fragile when AI enters a decision chain whose effects must be justifiable, bounded, and auditable in an opposable manner.

The paradox noted in the introduction takes on its precise meaning here. These organizations do not lack policies. They deploy systems that were not designed to be governed by those policies. Governance chases architecture and that race is not won by accumulating documents.

III. Three Structural Insufficiencies of Purely Procedural Governance

3.1 Retrospective Traceability Is a Reconstruction, Not a Proof

In a standard ML pipeline, the output of an inference is not, by default, accompanied by a complete and immutable reconstruction of the operational context that made that inference possible. What is typically preserved is a variable combination of inputs, outputs, timestamps, version identifiers, and application logs. This material is useful. But it does not guarantee a complete, stable, and legally robust reconstruction of the individual decision.

When one seeks to understand a particular decision after the fact, two families of tools appear. The first involves post hoc explainability. Methods such as SHAP or LIME can provide useful indications about local feature contributions or certain behavioral regularities of the model. But they are not the decision itself. Their value is real; their probative status must be handled with precision. The problem is not that they are useless. The problem is that they are too often mobilized as a substitute for a traceability that the system was never designed to produce.

The second family involves auditing available traces. But a log never contains anything other than what the system was designed to produce. In a heavily regulated environment, the distinction between faithful reconstruction of the execution context and interpretive approximation is not an academic subtlety. It determines the robustness of the audit, the qualification of an incident, and sometimes the very credibility of the compliance apparatus in front of a regulator who knows how to ask the right questions.

3.2 Inventory-Based Governance Remains Vulnerable to Shadow AI

Procedural governance implicitly rests on a strong assumption: the perimeter of systems to be governed is reasonably known. This assumption is becoming increasingly fragile. Data communicated by Larridin in early 2026 pointed precisely in this direction, indicating that a majority of organizations surveyed did not have a complete inventory of their AI applications and that the average number of deployed tools was already high. These figures do not constitute general proof on their own, but they do document the practical difficulty of the problem.

Precision is required here about what architectural governance can and cannot do. It can make detectable by construction the integration of non-compliant components within the institutionally integrated application perimeter. A component that does not respect integration invariants does not silently merge into the central architecture; it generates a composition anomaly or remains outside the institutional decision chain. It does not, however, eliminate lateral, opportunistic, or extra-system uses that constitute shadow AI in the broader sense. The difference remains decisive: it fundamentally changes the visibility regime of the problem, and the organization's position with respect to its risk becomes more defensible.

3.3 Under Operational Pressure, Procedure Degrades Faster Than Structure

Governance policies are applied in real organizations that is, under constraints of time, resources, and competing priorities. In these conditions, what is not constitutive of core business functioning is more easily deferred, deprioritized, or bypassed. Intermediate documentation, periodic risk reviews, certain cross-functional alignment steps undergo predictable degradation when an organization enters emergency mode. This is not a question of ill will. It is a question of organizational gravity.

An architectural property does not enjoy a higher moral status. It enjoys a different technical status. If an inference requires the effective persistence of the trace elements necessary for its reconstruction as a valid institutional event, the failure of that persistence is no longer a mere documentary shortfall. It becomes a failure of the normal execution regime. If the transition from a recommendation to an automated action is structurally constrained, circumvention requires a system modification, a more costly, more visible, and more traceable act than a simple procedural oversight.

The difference is not absolute. It is one of cost, visibility, and probability of circumvention. That is already considerable.

IV. Three Constitutive Properties of Architectural Governance

If one accepts that governance cannot rest exclusively on procedure in regulated systems with high stakes, it remains necessary to specify what architectural governance concretely requires. I propose here three minimal properties, not as a decorative catalog of best practices, but as a coherent system in which each element conditions the effectiveness of the other two.

Property 1: Constitutive traceability.

Each relevant inference or decisional transition must produce, as a normal condition of its execution, a stable record of the elements necessary for its reconstruction: relevant input data, version of engaged components, execution context, applied decision regime, produced result, confidence level, and useful correlation identifiers. The decisive point is this: the trace must not be conceived as an optional byproduct of the system, but as a property of its mode of functioning. A system should not be able to durably validate an inference as an institutional event without persistence of the trace elements necessary for its reconstruction.

Property 2: Operational qualification of the validity domain.

A query must not be processed as if the model were universally valid across the entire possible input space. The system must evaluate, at the point of execution, whether the input lies within the operational validity domain of the model being used. This qualification can take various forms: distance in the representation space, density estimation, uncertainty measurement, similarity score, business rules. The form matters less than the position: the validity boundary must be part of the inference pipeline, not relegated to a secondary or optional evaluation. A system that predicts outside its validity space without signaling it is not a poorly governed system. It is a system that lies about its own certainties.

Property 3: Structural separation of decision regimes.

The system must explicitly distinguish what falls within effective automation and what falls within recommendation subject to human validation. This distinction cannot depend solely on an organizational directive. It must be encoded in the design of flows, interfaces, permissions, validation events, and escalation mechanisms. Transforming a recommendation into an automated action should not be a usage drift; it should require an architectural decision, a perimeter modification, and an explicit reassessment of responsibilities. Organizations that delegate this boundary to a usage directive typically discover, at the first incident, that the directive did not survive the pressure.

V. Native Auditability and Event-Driven Architecture

The central proposition of this article is the following articulation: certain architectures make native auditability more accessible, more stable, and more economical than others. Among them, event-driven architecture associated with event sourcing patterns occupies a particular place — not because it would be the only possible path, but because it is one of the patterns most naturally compatible with this property.

In such an architecture, relevant state changes are represented as explicit events, timestamped, identified, and persisted in an append-only log. The current state of the system can be reconstructed as the projection of the sequence of events deemed relevant. This pattern is generally presented for its advantages in terms of decoupling, resilience, and scalability. It also possesses a virtue that is rarely exploited to its full extent in regulated AI systems: it brings governance closer to the very substrate of the system's functioning. The event is not only the unit of information; it is the unit of audit. The same mechanism that makes components coordinable makes the system governable.

In such a framework, a relevant inference can be represented as an event or as a transition enveloped in a sequence of correlated events. Its operational context, version dependencies, validation regime, and downstream effects become easier to link in a coherent reconstruction chain. This does not make the internal mechanism of a deep model transparent. The nuance introduced in §I remains intact. It does, however, make the traceability of its inscription within a governed process substantially more robust.

Furthermore, in a system strongly structured by events, a component that does not respect the expected schemas, validation invariants, or necessary persistence conditions becomes more difficult to silently integrate into the institutional flow. This eliminates neither parallel uses nor opaque third-party components. It reinforces the central system's capacity to distinguish what belongs to it from what is external to it.

VI. What This Approach Does Not Resolve

A serious thesis is also judged by the clarity of its limits. Here are four, stated without attenuation.

Architectural governance has a real upfront cost.

Designing explicit flows, maintaining event schemas, persisting relevant transitions, qualifying the domain at the point of execution, controlling transitions between recommendation and automation: all of this consumes time, storage, engineering discipline, and sometimes measurable latency. This cost is not negligible. The argument is not that it would be. It is that it is often lower, over the system's lifecycle, than the cumulative cost of governance reconstructed after the fact on an architecture that was never designed to be governable.

It does not replace procedural governance.

It makes procedural governance more credible and more effective. Documentation obligations, roles, arbitrations, responsibilities, committees, risk assessments, post-market surveillance plans, change management: all of this remains necessary. A governable architecture does not eliminate the need for human organization. It prevents that organization from being condemned to govern a system that technically escapes it.

Third-party foundation models remain partially opaque.

When a system integrates an external LLM via API, architectural governance can strongly frame inputs, outputs, call contexts, authorized uses, and human validation points. It does not make observable the entirety of the internal generation process. The native auditability of the integrating system is not the native auditability of the third-party model. This must be stated plainly, as ambiguity on this point is one of the most frequent sources of false confidence in current governance arrangements.

The decision/recommendation boundary becomes more complex in agentic systems.

In multi-agent architectures, certain intermediate decisions structure the option space available for the final decision, without always being exposed with the same legibility to the human user. The structural separation of decision regimes remains a valid objective. Its implementation becomes more delicate when the orchestration chain is itself dynamic, distributed, or adaptive. This is an open problem in the literature, and it would be premature to claim that architectural governance resolves it entirely in the current state of the art.

VII. Two Implementation Terrains

The value of the distinction between procedural and architectural governance is not purely speculative. It becomes particularly tangible in two types of systems developed within the Twingital Institute.

In the Sentinelle IA / PREDICARE program, the patient digital twin is modeled as a sequence of physiological, therapeutic, and contextual events. Each measurement, each prescription, each interaction with the healthcare system is represented as an immutable timestamped event, persisted in a state journal. The current state of the twin is calculated as the projection of this sequence. This choice was not motivated primarily by a documentary compliance objective, but by a requirement of correct modeling of clinical dynamics: events are here closer to clinical reality than frozen aggregates. It produces an important governability benefit: each predictive decision can be placed within a reconstructable chain of events, not by post hoc approximation, but because the system's structure was designed for this. One must nonetheless be explicit on this point: such an architectural choice carries neither automatic presumption of compliance nor substitution for MDR cycle requirements. For medical devices, post-market surveillance technical documentation falls under Annex III, while Annex XIV governs clinical evaluation and, for its Part B, PMCF. A more traceable architecture facilitates governability and can support the regulatory dossier; it replaces neither quality requirements, nor clinical evaluation, nor applicable documentary obligations.

In the ToxTwin V2.3 pipeline, the applicability domain is evaluated for each molecule submitted to the GINEConv OGB model, based on its distance to nearest neighbors in the space of molecular representation descriptors. This evaluation is not an optional post-processing tool; it is a component of the inference pipeline whose result conditions the presentation of the output. This choice was motivated by a precise technical observation: a model trained on low-molecular-weight organic compounds produces poorly interpretable confidence scores for structures chemically distant from its training space. Without prior qualification, the system would display high confidence on out-of-domain cases with the same interface as for cases close to the training corpus. This is a form of architectural disinformation. What this instance illustrates is more modest, and more useful, than a general

proclamation: operational domain qualification is integrable into an industrial prediction pipeline. It does not establish, on its own, the universal superiority of one applicability domain definition method over all alternatives.

One must resist the temptation to transform each coherent instance into a metaphysical truth. These two cases have value as illustrations of feasibility, not as general proof.

VIII. Practical Implications

The most important consequence is a shift in sequencing.

Most organizations still approach AI governance according to an implicit multi-phase logic: exploration, development, deployment, then framing. This sequence may have been tolerable for decision systems with low individual stakes. It becomes fragile when audit requirements, traceability obligations, and control over the automation regime must hold at scale, over time, and under real operational pressure.

The challenge is therefore not to add more rules or more committees. It is to bring into architectural decisions, much earlier, questions that are still too often treated as belonging exclusively to compliance: what must be traceable by construction, what must be bounded before execution, where exactly the boundary lies between assistance and automation, which components are entitled to produce system effects without traceable human validation. As long as these questions arrive after the stabilization of architectural patterns, governance remains condemned to chase systems already too free to be genuinely governed by the policies applied around them.

Conclusion

The industry does not need to oppose procedural governance and architectural governance. It needs to stop believing that the former can durably compensate for the absence of the latter.

Compliance frameworks, risk management reference systems, management standards, and documentary obligations are necessary. They structure responsibilities, organize oversight, enable audit, and constitute a common language between engineering, quality, compliance, and leadership. But they are not sufficient, on their own, to guarantee that a regulated AI system is effectively governable at the point where it acts, that is, at the moment of each individual decision, under operational constraint, within a perimeter that no one entirely controls.

The central question is therefore not merely: how do we govern the AI we deploy? It is more upstream, and more demanding: how do we design systems whose governability is part of the architecture, and not of the documentation added around it?

It is less a question of maturity than a question of sequencing. And in engineering as in medicine, sequencing errors are rarely spectacular at the outset. They become costly above all when it is already too late to pretend they were merely a detail.