FERZ, Inc.

Deterministic Governance Is Multi-Dimensional

Beyond Authorization in AI Systems

Why execution-time authorization, while necessary, cannot by itself deliver the deterministic governance that regulated environments require.

Edward Meyman
Founder & CEO, FERZ, Inc.
ferz.ai

March 2026

Abstract

Modern AI governance discussions increasingly emphasize execution-time authorization and action gating. These mechanisms are valuable and often necessary in regulated environments.

However, authorization alone cannot ensure that decisions are valid, consistent, or reproducible. Authorization answers the question “May this action proceed?” It does not answer the prior question: “Is the decision that produced this action correct under a formal governance model?”

Deterministic governance requires multiple architectural dimensions working together: governance ontology, constraint codification, deterministic decision logic, proof-carrying decision artifacts, and execution authorization. Each dimension addresses a distinct failure mode. No single dimension can substitute for the others.

This paper introduces the Multi-Dimensional Deterministic Governance Model, a framework for understanding how these components interact to produce reliable governance outcomes in AI-driven systems. It provides an evaluation framework that organizations can use to assess governance infrastructure against the full set of requirements, not merely the most visible one.

1. Introduction: The Governance Problem

AI systems increasingly participate in financial transactions, healthcare decisions, legal actions, and critical infrastructure operations. In each of these domains, the consequences of incorrect or unauthorized action are severe: regulatory sanctions, patient harm, fiduciary breach, or systemic risk.

These environments require governance systems that are deterministic (identical inputs produce identical outcomes), explainable (the basis for each decision can be reconstructed), auditable (independent parties can verify compliance), and enforceable (unauthorized actions are blocked before execution, not flagged after the fact). Recent taxonomic work [1] has formalized these distinctions, separating operational visibility, behavioral alignment, and decision authorization as fundamentally different governance problems.

Over the past two years, the governance conversation has shifted toward a specific and important mechanism: execution-time authorization. These systems evaluate whether a specific actor may take a specific action, in a specific context, before that action executes. This mechanism addresses a real gap. Most AI governance programs prior to this shift focused on data governance, model governance, and security governance, all of which operate either before or after the decision point.

The recognition that execution-time enforcement matters is a genuine advance. The error lies in treating it as the entire governance architecture.

Authorization is one dimension of deterministic governance. It is the enforcement dimension. But enforcement without formal decision logic is enforcement of informally defined rules. Enforcement without constraint codification is enforcement of whatever the upstream systems happen to declare. Enforcement without ontology is enforcement in a domain where the meaning of terms is unstable. And enforcement without proof-carrying decision artifacts is enforcement that cannot be independently verified.

This paper introduces a framework for understanding governance as a multi-dimensional architecture. It is not an argument against authorization. It is an argument that authorization operates correctly only when the other dimensions are present.

2. Authorization: An Essential Dimension

Execution-time authorization determines whether a proposed action may proceed. At the moment an actor (human, AI agent, automated workflow) attempts to execute a governed action, the authorization layer evaluates the request against applicable rules and returns one of a small number of outcomes: approve, deny, or escalate.

This pattern has deep roots in enterprise architecture. Payment authorization systems, API gateways, role-based access control, and network firewalls all implement some version of it. The pattern is well understood and operationally proven.

When applied to AI governance, authorization adds a capability that most existing governance frameworks lack: the ability to structurally prevent an action from executing.

Data governance can ensure that training data meets quality standards. Model governance can ensure that a model passes validation tests before deployment.

Security governance can ensure that only authenticated users access the system. None of these can block a specific action at the moment of execution based on the action's context, authority chain, and policy state.

Authorization fills that gap. It provides fail-closed enforcement at the execution boundary: if the authorization check cannot confirm that the action is permitted, the action does not proceed.

That is genuinely valuable. Organizations operating in regulated environments need this capability.

The question is not whether authorization matters. It does. The question is whether authorization alone constitutes deterministic governance.

It does not, for reasons that become clear when we examine what authorization depends on.

3. The Multi-Dimensional Governance Model

Deterministic governance is not a single mechanism. It is an architectural property that emerges from the interaction of five distinct dimensions, each addressing a different governance requirement.

Dimension	Core Question	Capability	Failure If Absent
Governance Ontology	What kinds of things exist in the governed domain?	Stabilizes meaning of actors, actions, objects, and authority	Policies are ambiguous; systems interpret rules inconsistently
Constraint Codification	How are policies expressed as machine-verifiable rules?	Enables deterministic evaluation, conflict detection, solver verification	Governance depends on informal interpretation
Deterministic Decision Logic	Does identical governed state produce identical outcomes?	Guarantees reproducibility and regulatory defensibility	Decisions vary run-to-run; audit is unreliable
Proof-Carrying Decisions	Can a third party verify that this decision is valid?	Produces replayable, cryptographic evidence of decision correctness	Audit records show occurrence, not validity

Execution Authorization	May this action proceed?	Prevents unauthorized execution; fail-closed enforcement	Unauthorized actions execute unchecked

Table 1. The five dimensions of deterministic governance. Each addresses a distinct failure mode. No single dimension can substitute for the others.

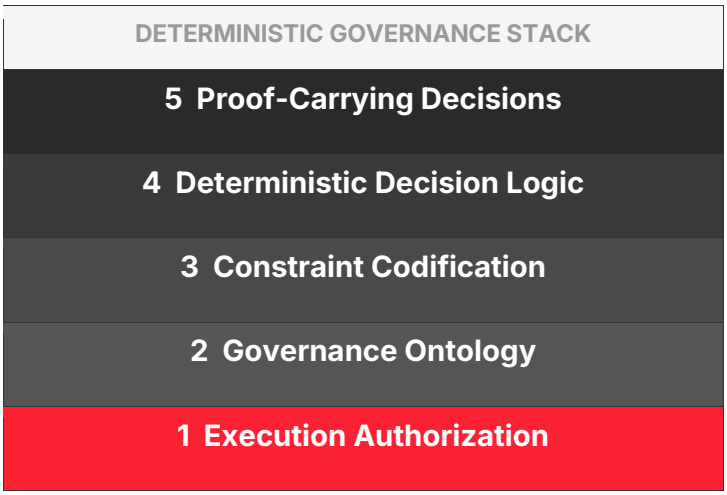

Figure 1. Operational view: authorization sits at the execution boundary; the upper layers determine correctness.

These dimensions can be viewed in two ways. As a logical dependency chain, ontology precedes everything: it defines the domain, constraints encode policy within it, decision logic evaluates constraints, proofs validate decisions, and authorization enforces outcomes. As an operational stack, authorization sits at the execution boundary and the upper layers supply the correctness that makes enforcement meaningful. Both views are accurate. The logical view explains why each dimension exists. The operational view explains where each dimension acts.

An organization may implement these dimensions in any order. But the governance guarantee is only as strong as the weakest dimension.

4. Governance Ontology

Before a governance system can evaluate whether an action is permitted, it must establish what kinds of things exist in the governed domain and how they relate to each other. This is the function of governance ontology.

A governance ontology defines, at minimum, actor classes and authority relationships (who may act, under what authority, with what delegation chains), action classes and

admissibility conditions (what kinds of actions exist, which are governed, which require special handling), object and target classes (what the action operates on, including sensitivity classifications and scope boundaries), policy objects and exception structures (how rules are organized, versioned, and scoped), and decision states (the possible outcomes a governance evaluation can produce).

Without a formal ontology, governance rules are ambiguous. The term “advisor” may mean different things in different policy contexts. The action “approve” may carry different authority requirements depending on the object class. The jurisdiction “US-VA” may trigger different constraint sets depending on whether the governing policy references state or federal authority.

Authorization systems that rely on a compact input schema (actor, action, context, consent) are performing ontological work implicitly. They assume that the upstream systems have already resolved these ambiguities and that the structured payload arriving at the authorization gate carries stable, unambiguous meaning.

That assumption may hold in simple environments. In complex regulated environments with multiple policy sources, overlapping jurisdictions, and evolving terminology, it frequently does not. Governance ontology is what makes it hold by construction rather than by hope.

In many governance systems, ontology is effectively implicit: meaning is derived from IAM roles, database schemas, and workflow definitions rather than formally specified. This causes semantic drift [2], where the same term carries different meaning in different system contexts without any mechanism to detect or resolve the inconsistency. A well-architected governance system treats ontology as an explicit, versioned component that feeds constraint codification and is referenced in decision artifacts, not as an emergent property of whatever the upstream systems happen to declare.

A critical design principle follows: authorization engines should consume ontology, not define it. If authorization is the layer that determines what terms mean, governance becomes dependent on integration context, inconsistent across deployments, and difficult to audit. Ontology must be stabilized upstream of enforcement.

5. Constraint Codification

Governance policies, as written by regulators, legal counsel, and compliance officers, are typically expressed in natural language. They contain conditional structures, exceptions, cross-references, and implied context. In most enterprises, relevant policies are distributed across identity systems, compliance platforms, workflow engines, and legal documents. A governance system that evaluates these policies must first transform them into machine-verifiable constraints.

Constraint codification is the process that transforms these fragmented sources into a coherent constraint regime. A codified constraint has an explicit condition (when does this rule apply?), an explicit action (what does the rule require, prohibit, or permit?), a priority (how does this rule interact with other rules that apply to the same input?), and a version (which version of the policy does this rule implement?).

Without codification, governance systems must rely on one of two fallback strategies. The first is to externalize constraint definition entirely, depending on whatever rules the customer's identity, compliance, and case management systems happen to expose. This strategy inherits upstream ambiguity: if the customer's policy sources are informally defined, inconsistently structured, or semantically unstable, the governance system enforces those weaknesses faithfully. The second is to use probabilistic or heuristic policy evaluation, which introduces non-determinism and makes audit-grade reproducibility impossible.

Deterministic governance requires a codification methodology: a repeatable process for transforming regulatory obligations into versioned, machine-verifiable, conflict-checked constraint sets. The output of codification is not merely a list of rules. It is a formally structured constraint regime that has been validated for completeness (all applicable rules are represented), consistency (no two rules produce contradictory outcomes for the same input), totality (every rule predicate returns true or false for any input), and fail-closed closure (every evaluation terminates in either a valid output or a governed denial).

Organizations that skip codification and proceed directly to enforcement are building a locked gate in front of an unlocked house. The gate may be non-bypassable. The rules it enforces may not be.

6. Deterministic Decision Logic

A deterministic governance system guarantees that identical governed state produces identical outcomes. This is not merely a desirable property. It is a precondition for regulatory defensibility, audit-grade reproducibility, and automated enforcement. Determinism is not an implementation preference. It is a governance requirement.

Decision logic is deterministic when the evaluation function is complete (it produces an outcome for every valid input), the evaluation function is non-contradictory (it never produces conflicting outcomes for the same input under the same constraint set), and the evaluation function is formally evaluable (its behavior can be verified through mathematical or logical analysis, not merely through empirical testing).

Determinism does not mean static behavior. A deterministic system responds to changes in its governing environment: when constraints change because a new regulation takes effect, outputs change accordingly. What determinism guarantees is that the system's behavior is fully determined by its inputs and governing constraints. For any given input-constraint pair, the output is fixed and reproducible. Change the inputs or the constraints and the output may change. Hold them constant and the output must not.

Systems that claim fail-closed behavior but lack deterministic decision logic face a structural problem: they cannot prove that the fail-closed property holds across all inputs. A system may fail closed on every input in the test suite and fail open on the first production input that exercises an untested code path. Deterministic decision logic, validated through formal methods (constraint solvers, SMT verification, or equivalent techniques), provides the mathematical guarantee that fail-closed behavior is a system property, not an empirical observation.

7. Proof-Carrying Decisions

Governance decisions must produce evidence. But the character of that evidence matters.

Most governance systems produce audit logs: records that a decision occurred, at a particular time, with a particular outcome. Logs are useful for forensic reconstruction. They answer the question: “What happened?”

Proof-carrying decisions answer a harder question: “Was the decision valid?”

A proof-carrying decision (PCD) is a cryptographically signed artifact that carries sufficient information for an independent third party to verify that the decision was correct under the governing constraint set. In a strong implementation, a PCD includes cryptographic hashes of the input and output, a complete record of every rule in the active constraint set and whether it was evaluated, its condition met, and any transformation applied, a version identifier and cryptographic hash of the constraint set itself (enabling exact replay), and an attestation of compliance-preserving equivalence.

The distinction between logs and proofs is not academic. A log asserts that a governance check occurred. A proof enables an auditor, regulator, or court to replay the decision independently and confirm that the same inputs, under the same constraints, produce the same outcome. This is the difference between describing governance and proving governance. Conformance standards for proof-carrying governance artifacts [5] formalize these requirements as machine-verifiable test criteria.

Authorization systems that produce sealed decision records with tamper-evident hashes are moving in the right direction. But a tamper-evident record of a decision is not the same as a replayable proof of decision validity. The hash proves that the record has not been altered. It does not prove that the decision was correct under a formally defined constraint regime. That requires the richer structure of a proof-carrying decision.

8. Execution Authorization in the Stack

With the other four dimensions in place, authorization performs its essential function: it translates governance logic into real-world enforcement.

Authorization sits at the execution boundary. It is the last governance checkpoint before an action takes effect. Its function is to enforce the outcome of the governance evaluation: permit actions that have been approved, block actions that have been denied, and route ambiguous or high-risk actions to supervised override pathways.

Authorization should generate or bind the enforcement evidence that regulators, insurers, and courts require: who attempted the action, what the governance outcome was, and what enforcement occurred.

Without authorization, governance cannot act. Ontology, constraints, decision logic, and proofs are structurally important but operationally inert unless the system can prevent unauthorized execution.

But without the other dimensions, authorization cannot guarantee correctness. It enforces whatever rules it receives. If those rules are informally defined, semantically unstable, or logically inconsistent, authorization enforces those deficiencies with the same mechanical reliability it would apply to well-formed constraints.

Authorization is the muscle of governance. It is not the brain.

9. Governance Completeness

A governance system achieves completeness only when all five dimensions are present and interacting.

Ontology defines the domain: what actors, actions, objects, and authority relationships exist.

Constraint codification encodes policy: how regulatory requirements, organizational rules, and contractual obligations are expressed as machine-verifiable constraints.

Deterministic decision logic evaluates constraints: identical governed state produces identical outcomes, verifiable through formal methods.

Proof-carrying decisions validate outcomes: cryptographic artifacts enable independent replay and verification of decision correctness.

Execution authorization enforces outcomes: unauthorized actions are blocked before execution, with sealed enforcement evidence.

Missing dimensions create governance gaps that no amount of strength in the remaining dimensions can close. Each missing dimension produces a distinct failure mode [4] with observable symptoms that regulators, auditors, and insurers can detect.

Missing Dimension	Consequence	Observable Symptom
Ontology	Rules are interpreted inconsistently across contexts	Same action approved in one context, denied in another, with no formal basis for the difference
Codification	Enforcement depends on informally defined upstream rules	Governance quality varies by customer maturity; system cannot detect policy conflicts
Decision logic	Outcomes vary for identical inputs	Audit reveals inconsistent decisions; regulator cannot reproduce results
Decision proofs	Decisions are auditable but not verifiable	Logs show what happened; no one can prove the decision was correct

Authorization	Unauthorized actions execute unchecked	Governance is advisory; system cannot prevent prohibited actions

Table 2. Governance gaps created by missing dimensions.

10. Implications for AI Systems

The multi-dimensional governance model is not specific to AI. Any system that makes consequential decisions under regulatory constraints benefits from governance completeness. But AI systems make the requirements more urgent for four reasons.

Velocity. AI systems generate actions faster than human supervision can review. Governance systems must operate at machine speed, which means enforcement must be structural (built into the execution path) rather than procedural (dependent on human review before each action).

Autonomy. AI agents increasingly operate with delegated authority, making decisions and taking actions without per-action human approval. Governance systems must ensure that delegated authority is bounded, that boundaries are enforced at runtime, and that every exercise of authority produces verifiable evidence.

Opacity. The internal reasoning of large language models and other AI systems is not directly interpretable. Governance systems cannot rely on understanding why the AI proposed an action. They must evaluate whether the action is permitted under the governing constraint regime, regardless of the AI's internal reasoning process. This makes formal constraint evaluation (not model interpretability) the operationally relevant governance mechanism.

Scale of delegation. AI systems can operate under delegated authority at a scale and frequency that human governance mechanisms were never designed to supervise. A single AI agent may exercise delegated authority thousands of times per hour across multiple jurisdictions. Governance infrastructure must match that scale without degrading the rigor of each individual evaluation.

Organizations deploying AI in regulated environments will face increasing pressure from regulators, insurers, and boards to demonstrate governance completeness across all five dimensions. Systems that provide only authorization will face the question: "You can prove that you checked. Can you prove that the check was correct?"

11. Evaluating Governance Infrastructure

The multi-dimensional model provides a neutral evaluation framework. Organizations assessing governance platforms can ask five diagnostic questions, one for each dimension.

1. Ontology: What formal model defines the actors, actions, objects, and authority relationships in your governed domain? How is semantic stability maintained across policy versions and jurisdictions?

2. Codification: How are regulatory requirements and organizational policies transformed into machine-verifiable constraints? What methodology ensures completeness, consistency, and conflict-free rule sets?

3. Decision logic: Does identical governed state always produce identical governance outcomes? What formal verification supports this guarantee?

4. Decision proofs: Can an independent third party replay a governed decision and verify its correctness? What artifact structure enables this verification?

5. Authorization: Is the enforcement boundary non-bypassable for governed workflows? What happens when an evaluation is ambiguous or a required input is missing?

A governance system that answers all five questions with verifiable evidence provides governance completeness. A system that answers only the fifth question provides a gate, not governance completeness.

This evaluation framework does not prescribe a specific implementation. Organizations may satisfy each dimension through different architectural choices. The framework asks whether each dimension is addressed, not how. For operational procurement guidance applying these principles, see [3].

12. Conclusion

Authorization is a powerful enforcement mechanism. Its importance should not be understated. A governance architecture that cannot prevent unauthorized actions from executing has a structural gap that no other capability can compensate for.

But authorization is one dimension of a five-dimensional problem.

Deterministic governance requires governance ontology to stabilize meaning, constraint codification to formalize policy, deterministic decision logic to guarantee reproducibility, proof-carrying decisions to enable verification, and execution authorization to enforce outcomes.

Organizations deploying AI in high-consequence environments should evaluate their governance infrastructure against all five dimensions. Strength in one dimension does not compensate for absence in another. The governance guarantee is only as strong as the weakest dimension.

> *Authorization governs the exit door. Deterministic governance governs the legality of the decision that reaches the door, the authority structure behind it, and the proof that the outcome is valid.*

The multi-dimensional model provides the framework for making that distinction precise, evaluable, and actionable.

References

[1] Meyman, E. (2026). A Taxonomy of AI Governance Approaches: Distinguishing Visibility, Alignment, and Authorization. Version 1.5. FERZ, Inc. Zenodo. https://doi.org/10.5281/zenodo.18275969

[2] Meyman, E. (2026). Versioned Meaning: How to Make Ontologies Audit-Stable. Version 1.4. FERZ, Inc. SSRN: https://ssrn.com/abstract=5918182. Zenodo: https://doi.org/10.5281/zenodo.18328587

[3] FERZ, Inc. and Meyman, E. (2026). The Enterprise AI Governance Buyer's Guide. Version 3.1. Zenodo. https://doi.org/10.5281/zenodo.18002693

[4] Meyman, E. (2026). Governance Laundering: A Taxonomy of Failure Modes in AI Compliance Architectures. Version 1.0. FERZ, Inc. Zenodo. https://doi.org/10.5281/zenodo.18746522

[5] Meyman, E. (2025). Four Tests Standard (4TS) for Deterministic AI Governance. Version 1.0.2. FERZ, Inc. https://github.com/edmeyman/4ts-standard

How to Cite This Work

Meyman, Edward (2026). Deterministic Governance Is Multi-Dimensional: Beyond Authorization in AI Systems. Version 1.0, March 2026. FERZ, Inc. Zenodo. https://doi.org/10.5281/zenodo.18902167

About the Author

Edward Meyman is Founder and CEO of FERZ, Inc. (McLean, VA), which builds deterministic governance infrastructure for AI systems in regulated environments. FERZ's architecture implements the multi-dimensional governance model described in this paper across healthcare, financial services, government, and legal verticals. The Four Tests Standard (4TS), published as a vendor-neutral open standard, provides conformance criteria for deterministic governance systems.

Contact: ferz.ai

ORCID: 0009-0008-8012-6100