ENTERPRISE ARCHITECTURE · GOVERNANCE

Frameworks Describe. Something Must Govern.

Why enterprise architecture frameworks need an authority layer to stay true

ABSTRACT

Enterprise architecture frameworks — TOGAF, Zachman, BIZBOK, ArchiMate, SAFe — define the intent, structure, and governance that align technology with the business. They were built to *describe* architecture, not to *enforce* it. In environments shaped by automation, decentralization, and continuous change, descriptions drift: the documented architecture and the deployed reality diverge, and no framework was designed to keep them reconciled. This paper argues the missing element is not another framework but an **authority layer** — an Infrastructure Operating Model (IOM) that holds declared architectural intent as the binding reference against which reality is judged. The IOM does not replace frameworks or observe its way to truth. It gives every framework a single authoritative account of what the architecture is *supposed* to be, so that conformance, reuse, and governance become continuous properties rather than periodic attestations.

1. Frameworks were built to describe, not to enforce

Enterprise architecture frameworks do something genuinely valuable: they give an organization a disciplined way to express what its architecture should be. TOGAF supplies method; Zachman supplies a taxonomy of viewpoints; BIZBOK ties business capability to realization; ArchiMate gives the whole thing a modeling language; SAFe and similar delivery frameworks connect it to how work actually ships. Each addresses a different concern, and most enterprises run several at once.

What none of them was designed to do is *operate*. A framework is a way of describing and governing architecture at the moments humans convene to review it — a design board, a planning increment, an annual reassessment. Between those moments, the architecture keeps changing: pipelines deploy, teams refactor, services are stood up and retired. The framework's description does not change with it. The map and the territory part company, quietly, continuously.

> ***A framework tells you what the architecture should be. It has no way of knowing whether that is still true.***

2. The real failure is drift, and drift is a question of authority

It is tempting to diagnose the problem as fragmentation — each framework keeping its own partial picture, the pictures disagreeing, trust eroding. That is a real symptom. But the deeper failure is **architectural drift**: the divergence between architectural intent and deployed reality over time. And drift is not, at root, an observation problem. You can observe perfectly and still have no answer to the question that matters.

That question is: when the deployed reality differs from the documented architecture, *which one is wrong?* Sometimes reality has drifted from a still-valid intent, and reality must be corrected. Sometimes intent is stale and reality reflects a sanctioned change that was never written down, and the intent must be updated. Observation can tell you the two disagree. It cannot tell you which one carries authority. Resolving drift requires knowing what was *declared* — what the architecture is legitimately supposed to be, and on whose authority — not merely what currently exists.

> ***Observation establishes what exists. Only declared intent establishes what is legitimate. Drift is the gap between them, and it is settled by authority, not by measurement.***

3. The missing element is an authority layer, not another framework

This is why the answer is not a sixth framework, nor a better modeling notation, nor a more complete repository. Adding descriptive capability does not resolve a question of legitimacy. What architecture practice is missing is an **Infrastructure Operating Model (IOM)**: an authoritative layer that holds

declared architectural intent — capabilities, constraints, ownership, approved patterns — as the binding reference every framework and every change is judged against.

The distinction from a live repository or a digital twin is exact. A twin maintains an accurate representation of what infrastructure *is*. An IOM maintains the authoritative account of what infrastructure is *supposed to be*, and adjudicates the difference. The first is assembled from observation; the second is declared and then enforced. No increase in the fidelity of the representation ever produces that authority — authority is a different kind of fact, and it has to be stated.

What the IOM does for architecture practice

Concretely, an IOM gives enterprise architecture four things its frameworks cannot supply on their own. It holds a **single authoritative account of architectural intent** that multiple frameworks can each interpret through their own lens without maintaining competing versions of the truth. It treats **reuse as a governed, living property** — approved Architecture Building Blocks remain authoritative constructs that variation is measured against, not design-time artifacts that decay the moment they are published. It makes **conformance continuous**, evaluating change against declared intent before deployment rather than discovering nonconformance at the next review. And it provides **governance with an evidentiary basis**: exceptions are explicit, scoped, and time-bound against a known baseline, rather than negotiated against whatever picture each party happens to hold.

4. Frameworks stay intact; their execution becomes continuous

None of this displaces the frameworks. An IOM encodes no framework's doctrine, requires no framework to be modified, and introduces no competing model of architectural truth. It sits beneath all of them as a shared reference, so that each can keep doing what it does well while drawing on one consistent account of declared intent.

The relationship is straightforward in each case. For method frameworks such as TOGAF, the IOM keeps the baseline meaningful between lifecycle phases and lets Architecture Building Blocks persist as governed constructs rather than documents. For taxonomy frameworks such as Zachman, it keeps the classification anchored to authoritative intent so the taxonomy does not drift out of relevance. For business architecture such as BIZBOK, it preserves traceability from business capability to its sanctioned technical realization, surfacing divergence early. For modeling languages such as ArchiMate, it distinguishes the *intended* architecture from the *observed* one and supplies a stable reference for regenerating views. And for delivery and security frameworks, it turns post-deployment controls into architectural guardrails that evaluate change against intent before it ships.

The relationship is most concrete with TOGAF, whose lifecycle discipline and emphasis on reuse actually *raise* the need for an authority layer: organizations that apply the Architecture Development Method rigorously are often the first to hit the limits of framework-only execution, because their reuse and governance expectations exceed what episodic architectural truth can sustain. The table below

maps each ADM phase to the execution gap that opens without an operating model, and the capability an IOM supplies to close it.

TOGAF ADM Phase	Execution Gap Without an IOM	IOM Execution Capability
Preliminary	Principles exist only as policy statements	Principles encoded as enforceable structural constraints
A — Architecture Vision	Vision decays after approval	Vision bound to observable infrastructure boundaries
B — Business Architecture	Ownership not reflected in runtime systems	Ownership and accountability continuously encoded and observable
C — Data Architecture	Data sprawl invisible between review cycles	Continuous validation of data location and flow
C — Application Architecture	Dependency drift accumulates silently	Authoritative, continuously reconciled dependency model
D — Technology Architecture	Standards enforced manually or retrospectively	Blueprint-derived enforcement of approved patterns
E — Opportunities & Solutions	Transition plans diverge from execution	Real-time visibility into deviation and convergence
F — Migration Planning	Roadmaps disconnected from live state	Roadmaps grounded in current, authoritative reality
G — Implementation Governance	Governance occurs after risk is introduced	Preventative, evidence-based architectural control
H — Architecture Change Mgmt	Change detected late and incompletely	Continuous change detection and governed evolution

Table 1. TOGAF ADM phases mapped to the execution gap that opens without an operating model and the capability an IOM supplies. In each case TOGAF remains the architectural authority; the IOM supplies the execution capability that sustains it over time.

> ***Frameworks remain the language of architecture. The IOM is the authority that keeps that language true.***

5. Why this matters now

For most of enterprise architecture's history, the gap between description and execution was bridged by people. Architects carried the real state of the estate in their heads and reconciled it against the frameworks at review time. Continuous delivery already strained that arrangement. Autonomous and AI-driven change breaks it: systems now alter the architecture faster than any review cadence can inspect, and they do so without the architect's implicit sense of what was legitimately intended.

In that setting, an architecture practice that relies on periodic human reconciliation is governing a system it can no longer see in time. The choice is not between frameworks and an operating model; it is between architecture as a set of documents that age between reviews and architecture as a trusted decision-making system that stays current because something authoritative is keeping it current. The IOM is what makes the latter possible.

6. Conclusion

Enterprise architecture frameworks define intent, structure, and governance, and they do it well. They were never meant to function as execution systems, and faulting them for drift mistakes their purpose. The missing element is an authority layer: an Infrastructure Operating Model that holds declared architectural intent as the binding reference reality is reconciled against.

With that layer in place, frameworks remain intact and their execution becomes continuous. Conformance, reuse, and governance stop being episodic attestations and become durable properties of a living system. Architecture is restored to what it was always supposed to be — not a record of past decisions, but a trusted authority over present ones.

This is a category paper in the IOM Standard library. It establishes the conceptual relationship between enterprise architecture frameworks and the Infrastructure Operating Model; it does not restate the normative requirements of the IOM Standard itself. The IOM Standard is an open, vendor-neutral specification — learn more at theIOM.org.