AIOPS · OBSERVABILITY · AI GOVERNANCE

AIOps Is an Operating-Model Problem

AIOps the way it was meant to work — what it becomes when it consumes an authority layer instead of inferring one

ABSTRACT

AIOps has spent a decade fighting the same complaints: too many alerts, too little trust, correlation that reads as guesswork, root-cause analysis that points everywhere. The usual diagnosis is that the algorithms need to improve. They do not. AIOps reasons over telemetry, and telemetry records what happened — never what was *intended* or *permitted*. With no authoritative account of intent to reason against, even excellent models produce probabilistic output about a system whose correct state is undefined. This paper makes the constructive case: an **Infrastructure Operating Model (IOM)** supplies the missing ground truth, and AIOps consumes it. The IOM does not sit on top of AIOps; it sits above it as the authority layer, and AIOps becomes the analysis layer that finally has a reference frame. Grounded this way, correlation becomes intent-relative, noise collapses, and root cause becomes deterministic — not because the models got smarter, but because they were given something true to reason against.

1. The complaint that never goes away

Every generation of AIOps inherits the same reputation: powerful in the demo, noisy in production. Alert volumes stay high, on-call teams keep tuning thresholds, correlation surfaces plausible-but-wrong groupings, and root-cause analysis offers a ranked list of suspects rather than an answer. Each cycle, the proposed fix is better math — richer models, more training data, tighter baselines. Each cycle, the improvement is real but bounded, and the core complaint survives.

The persistence of the complaint across a decade of genuine algorithmic progress is the clue. When better models keep failing to dissolve a problem, the problem is usually not in the models. It is in what they are given to reason over.

> ***When a decade of better algorithms doesn't fix the complaint, the complaint isn't about algorithms.***

2. The boundary, briefly

AIOps reasons over telemetry — logs, metrics, traces, events, flows. Telemetry is a faithful record of what happened, when, and how often. It contains no record of what the system was *intended* to do, which connections are deliberate versus accidental, or which states are *permitted*. So when AIOps asks whether a behavior is anomalous, it can only answer relative to a statistical baseline: is this unusual compared to the recent past? It cannot answer the question operators actually care about: is this *wrong*?

Those are different questions, and the gap between them is the source of the noise. A surge in east-west traffic is statistically unusual and may be entirely legitimate; a quiet, in-baseline connection may be a serious policy violation. Without an authoritative account of intent, AIOps cannot tell the two apart, so it flags both, or neither, and asks a human to decide. This boundary is argued in full in the companion papers *Why AI Needs an Infrastructure Operating Model* and *Security Is an Operating-Model Problem*; here it is only the setup. The point of this paper is not that AIOps lacks authority — it is what AIOps becomes once something else supplies it.

3. The orientation correction: IOM above, AIOps below

It is tempting to say AIOps should be "built on top of" an operating model, but that phrasing inverts the architecture. The IOM is the authority layer; it sits *above* execution and analysis, holding declared intent as the binding reference. AIOps sits *below* it, alongside the other signal and analysis systems, and *consumes* the model as ground truth. The IOM is not a feature AIOps acquires; it is the reference frame AIOps reasons within.

The relationship is consume-and-defer, not stack-on-top. AIOps reads the IOM's model of intent, ownership, dependencies, and allowable states, and evaluates its telemetry against that model. It does

not extend, override, or become the model. When AIOps and the IOM disagree about what a system should be doing, the IOM is right by definition — that is what makes it the authority layer. AIOps contributes analysis; the operating model contributes truth.

> ***The IOM does not sit on top of AIOps. It sits above it. AIOps consumes the authority layer; it never becomes one.***

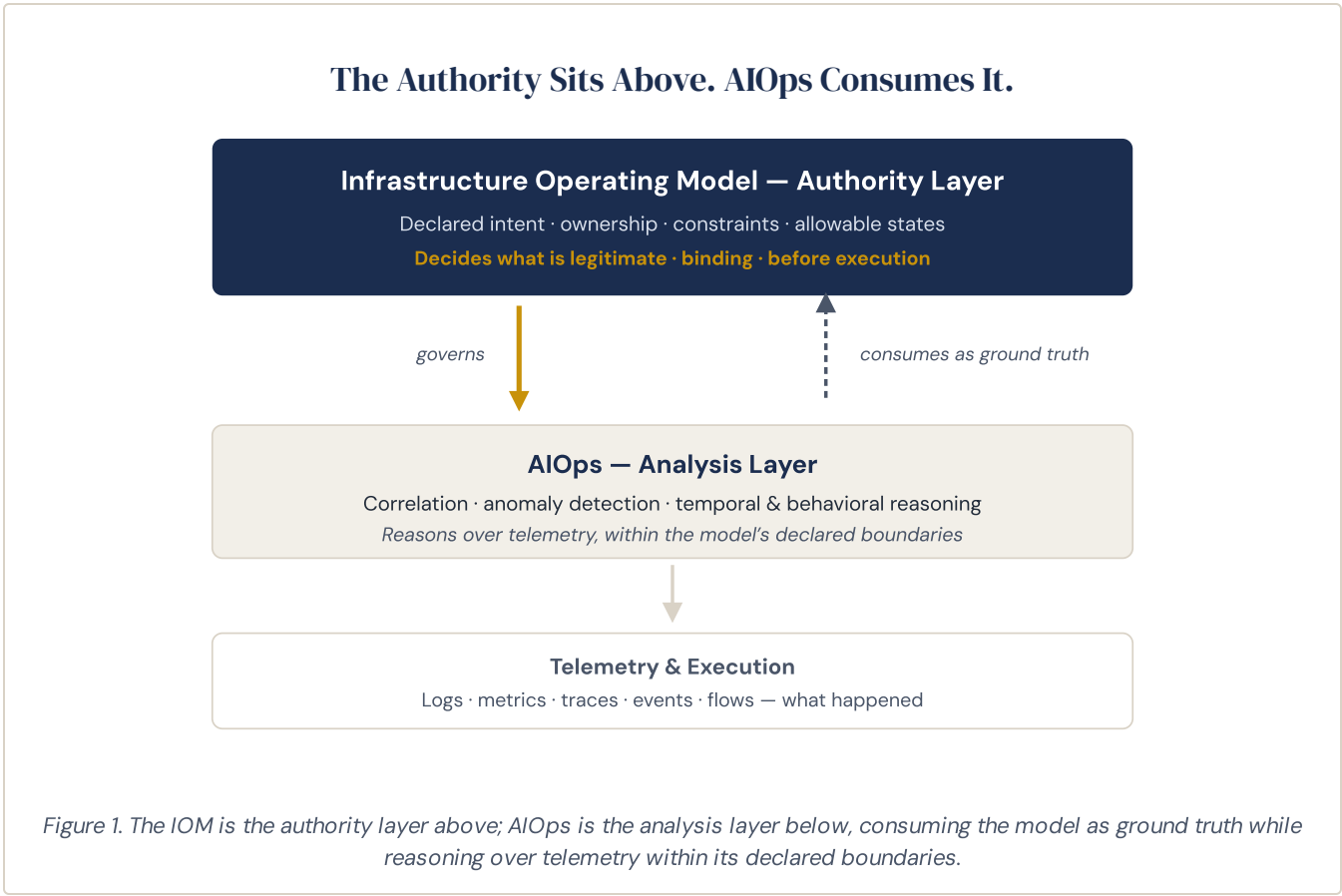

Figure 1. The IOM is the authority layer above; AIOps is the analysis layer below, consuming the model as ground truth while reasoning over telemetry within its declared boundaries.

4. Where intent comes from: ratification, not discovery

Before the payoff, one objection has to be answered, because every technical reader raises it: if AIOps infers and the IOM declares, isn't declaration just inference with a manual step bolted on? Both start from observed state, the argument goes, so both are ultimately guessing — the IOM merely makes a human rubber-stamp the guess. The objection is fair, and answering it is what separates an operating model from a better discovery tool.

The difference is in what happens to discovered state. A discovery tool reads the environment and stops at a reading: *this subnet exists, this security group is attached, this route is present*. The reading is the output. An IOM reads the same environment and treats it as a *proposal* — here is what your infrastructure currently implies about its own intent — and then puts that proposal through a step discovery tools do not have: an architect **ratifies** it. The architect accepts, corrects, or rejects each implied constraint, and what survives is no longer a reading of what exists. It is a declaration of what is

allowed to exist. Observation supplies the candidate; human judgment supplies the authority. That ratification step — not the observation that precedes it, and not any automation that follows — is the act that creates legitimacy.

This is why an IOM does not require a greenfield environment or a blank-page blueprint. Intent can be authored directly for new patterns, but in existing estates it is more often *derived*: the model imports running infrastructure, elevates the implicit intent embedded in how it was built, and presents it for ratification — converting "what exists today" into "what is governed going forward." And once intent exists, reconciliation keeps it honest: every drift between declared intent and running state forces a decision — correct reality, or update the intent — so the intent layer stays live rather than decaying into stale documentation. Discovery tools have no such forcing function, which is precisely why their picture rots. The full adoption mechanics of this — import, model, blueprint, enforce — are developed in the companion paper *From Tool Aggregation to Infrastructure Authority*; what matters here is the distinction: discovery ends at a reading, an IOM turns the reading into a ratified declaration.

> ***Discovery asks "what is configured?" and stops. An IOM asks "what is configured — and shall it become what is allowed?" The discovered state is identical; the ratification is the entire difference.***

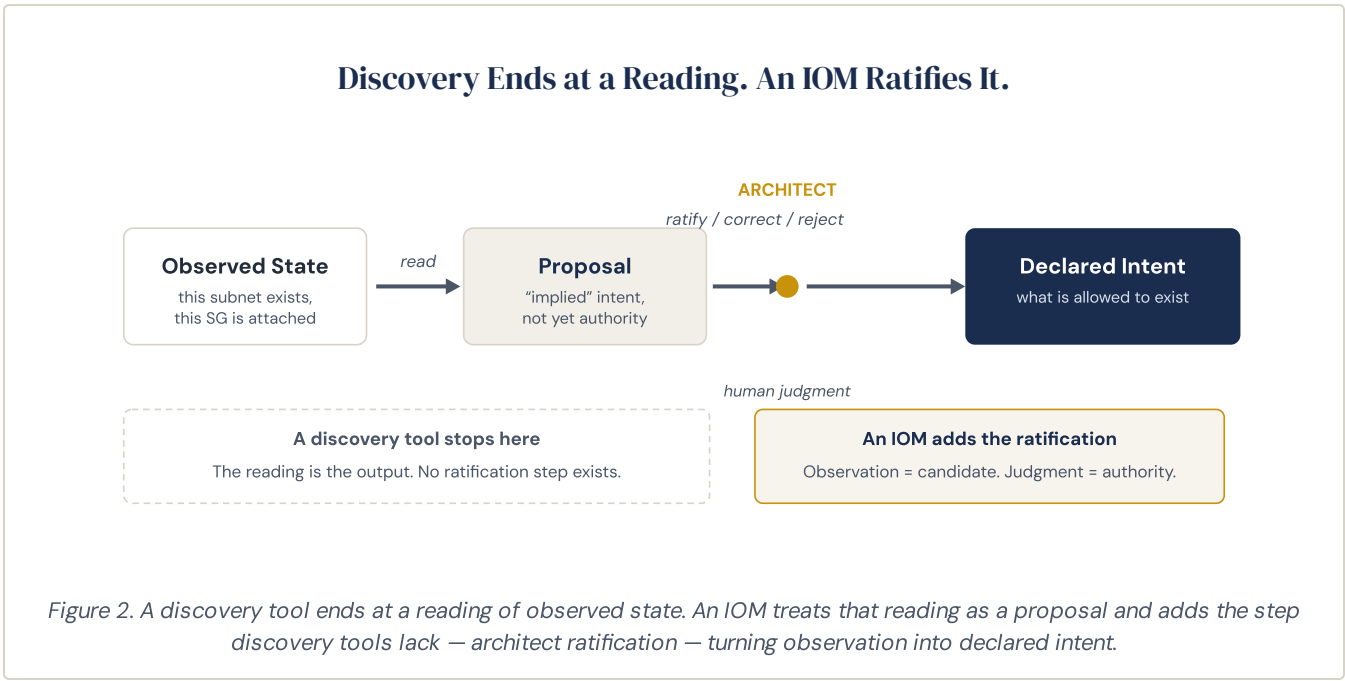

Figure 2. A discovery tool ends at a reading of observed state. An IOM treats that reading as a proposal and adds the step discovery tools lack — architect ratification — turning observation into declared intent.

5. What correlation becomes

Grounded on an operating model, correlation stops being statistical co-occurrence and becomes intent-relative reasoning. Today, AIOps groups events that tend to appear together and infers a relationship; the inference is probabilistic and breaks whenever the architecture shifts. With an authoritative dependency model to read from, AIOps no longer infers which components are related —

it *knows*, because the relationships are declared. Correlation shifts from "these alerts statistically cluster" to "these alerts lie along a known dependency path with a known owner."

The practical effect is that correlation becomes explainable and stable. Explainable, because the grouping traces to a declared relationship a human can inspect rather than a latent statistical artifact. Stable, because when the architecture changes, the model changes with it — the correlation logic does not silently decay between retrainings. The model does the structural reasoning; AIOps does the temporal and behavioral reasoning on top of it. Each does what it is actually good at.

6. What noise becomes

Alert noise is, overwhelmingly, the cost of evaluating signals without intent. A platform that cannot tell legitimate-but-unusual from illegitimate-but-quiet has no choice but to over-alert and let humans filter. That human filtering — triage — is the real expense of AIOps, and no amount of model tuning removes it, because the missing information is not in the telemetry to be learned.

The failure has a recognizable shape in the field. A large enterprise stands up a leading event-correlation platform and wires the full telemetry estate into it — logs, metrics, traces, flows, and the alert streams of a dozen monitoring tools. The platform performs exactly as designed: it correlates aggressively and compresses a flood of raw alerts into a far smaller set of consolidated incidents. The dashboards look transformative. And then the program stalls short of the outcome that was promised, for a reason the correlation engine cannot fix — the consolidated incidents are still descriptions of what happened. The platform ingested everything except the one thing that would have let it judge legitimacy, and that thing — declared intent — was never available to ingest, because no tool in the estate held it. The deployment did not underperform its category. It reached the ceiling of the category.

This exposes the quiet mismeasurement at the heart of AIOps value. Correlation is measured by how much it reduces *volume* — ten thousand alerts compressed to fifty incidents reads as a ninety-nine-percent win. But volume was never the problem worth solving. The problem is *uncertainty about legitimacy*, and correlation does not reduce it at all: all fifty surviving incidents are still "things that happened," and not one of them is yet established as "a thing that should not have." Compression changes how many items a human must inspect; it does not change the question they must ask of each, which remains "is this allowed?" — the one question the telemetry cannot answer. Reducing volume while leaving legitimacy-uncertainty untouched is why better correlation keeps producing fewer, cleaner alerts that are no more trustworthy than the many it replaced.

> ***Correlation reduces the volume of signals, not the uncertainty about which ones are legitimate. Compressing ten thousand alerts into fifty is a quantity win and a zero on the only question that matters: should any of the fifty be happening at all?***

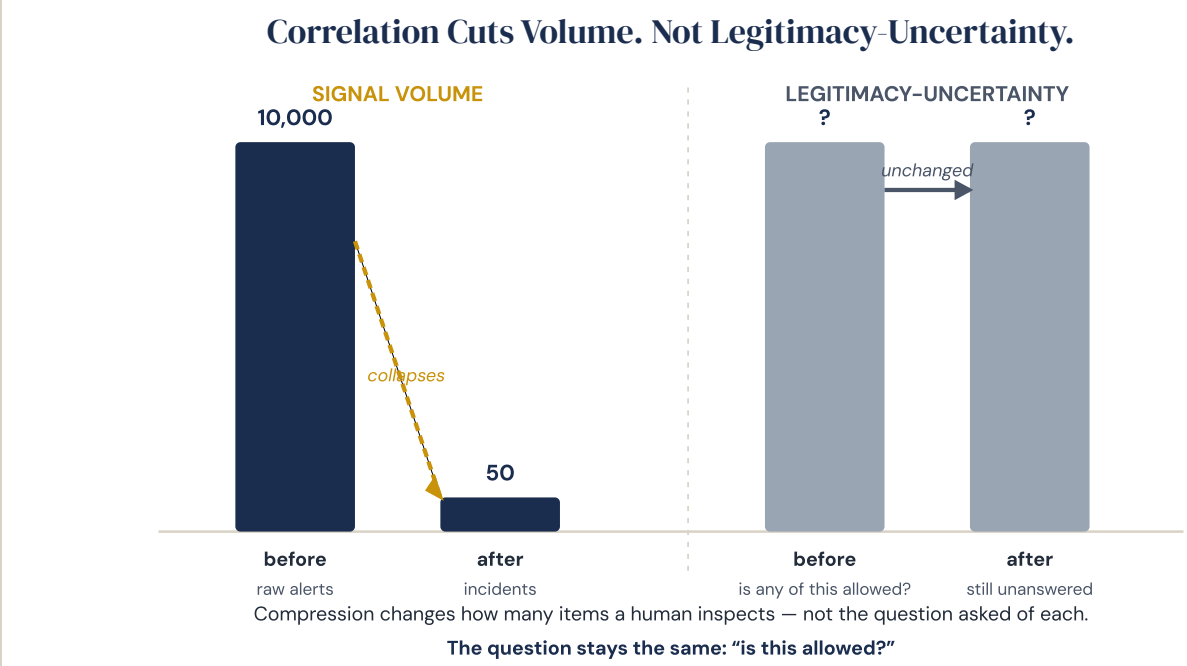

Figure 3. Correlation collapses signal volume dramatically while legitimacy-uncertainty stays flat. Compression changes how many items a human inspects, not the question asked of each.

When telemetry is reconciled against declared intent, the filter moves upstream and becomes deterministic. Behavior that conforms to intent is recognized as expected and never raised; deviation from intent is immediately meaningful and raised with context — the violated constraint, the owner, the blast radius. The question shifts from "is this statistically unusual?" to "is this allowed?", and only the second question produces alerts worth a human's attention. Noise does not fall because the model got more selective; it falls because most of what used to be alerted was never a deviation from intent in the first place.

> ***Triage is the cost of alerting without intent. Reconcile telemetry against declared intent and the filter moves upstream, where it can be deterministic instead of human.***

7. What root cause becomes

Root-cause analysis fails today because it runs on inferred structure. When the dependency graph is reconstructed probabilistically from observed traffic, causal reasoning over it inherits the uncertainty: the tool can rank likely causes but cannot prove one, because the graph it reasons over is itself a guess. This is why root-cause output so often reads as a list of suspects rather than a determination.

On an authoritative model, the dependency and ownership structure is declared, not inferred, so causal reasoning runs over ground truth. Impact propagation becomes deterministic — a failure's blast radius follows known dependencies rather than estimated ones — and the analysis can state not just

what is correlated with the incident but what, structurally, could and could not have caused it. AIOps still does the work of tracing the event timeline; it simply does it against a map that is true.

8. The proof already exists: intent-based networking

This is not a thought experiment. One infrastructure domain already rebuilt itself around exactly this separation, and it worked. Networking spent decades managing devices the way AIOps manages systems today — box by box, configure-monitor-troubleshoot, with correctness inferred from observed behavior. Then it reorganized around declared intent, and the result became a recognized category: intent-based networking.

The architecture is the one this paper argues for, in miniature. Intent-based networking runs a three-function loop — **translation** (capture declared intent and translate it into validated policy), **activation** (push that policy into the infrastructure), and **assurance** (continuously verify that the running network matches the intent). The decisive word is *declarative*: the operator states what must be true — this segment is isolated, this traffic class is guaranteed — rather than scripting how each device achieves it. Intent is held as a first-class artifact, separate from the per-device configuration that implements it. That is the intent-versus-implementation distinction, realized in production.

The assurance function is the part AIOps should study most closely, because it is config-level reconciliation done right. Cisco's Network Assurance Engine reads declared policy from the controller and configuration and dynamic state from each device, builds a formal model of the network, and continuously verifies that observed behavior conforms to operator intent — flagging violations before they cause outages. The earlier vendor-neutral pioneers made the same move across heterogeneous hardware: Apstra generated mathematically verified-correct configuration from declared design; Forward Networks and Veriflow built formal models that check the running network against intended state. Where a discovery tool would report the configuration as found, these systems verify it against what was declared — deterministically, at the config level, continuously.

> ***Networking already solved this. It stopped inferring correctness from device state and started declaring intent and reconciling against it. The model works in production — the unfinished work is that it stops at the edge of the network.***

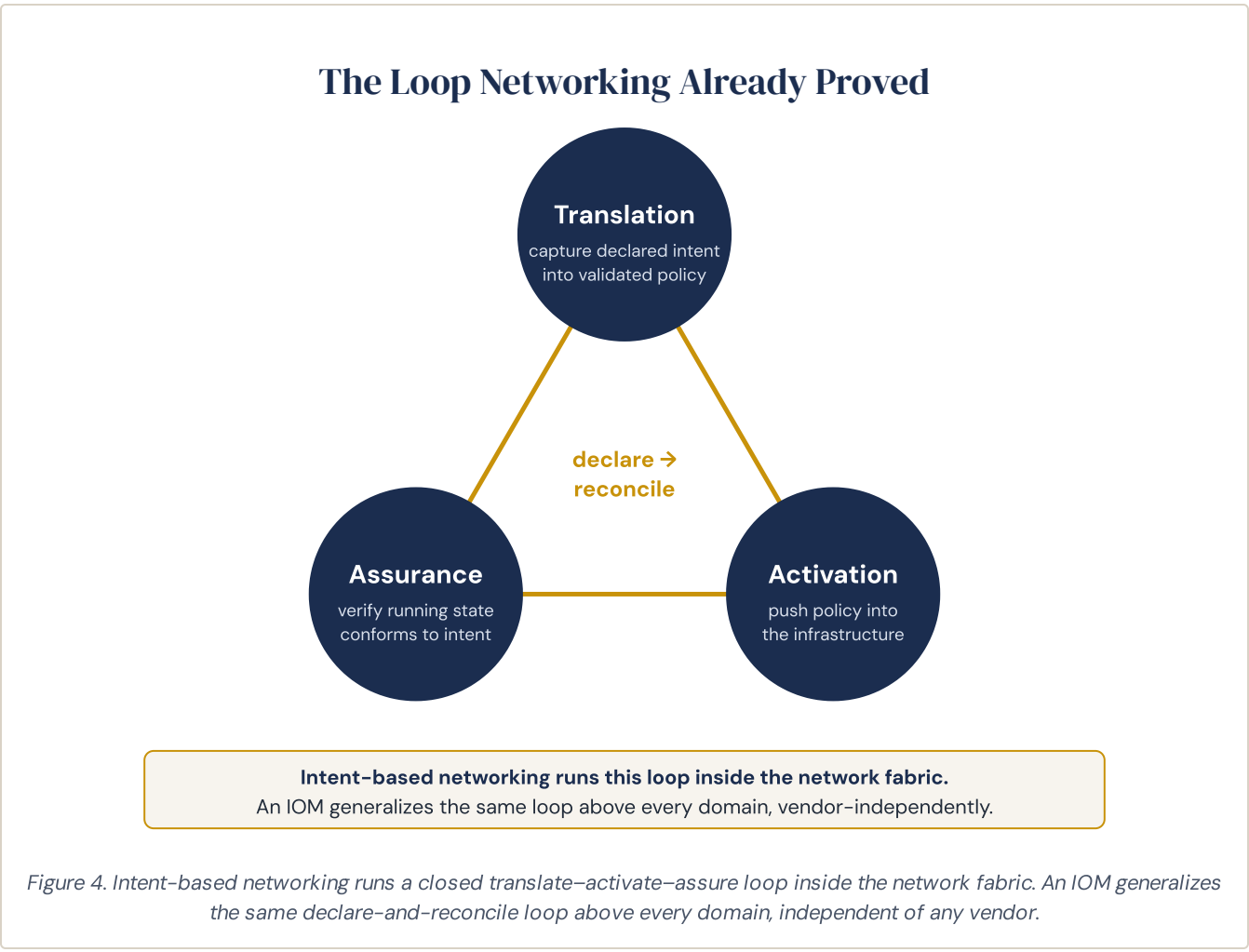

Figure 4. Intent-based networking runs a closed translate–activate–assure loop inside the network fabric. An IOM generalizes the same declare-and-reconcile loop above every domain, independent of any vendor.

The limit of intent-based networking is not its model but its *scope*. It governs the network fabric and, in its dominant implementations, is bound to a single vendor's controllers and hardware. The declared intent that matters to an enterprise does not stop at the network: it spans compute, identity, security, data placement, and cloud, and most of it lives in domains intent-based networking never sees. Networking proved the loop — declare, activate, assure — is correct and operable. An Infrastructure Operating Model is that same loop generalized above every domain and made independent of any one vendor's fabric. AIOps grounded on an IOM is what intent-based assurance looks like when it is no longer confined to the network.

9. The division of labor

None of this diminishes AIOps. It relocates it to the role it is genuinely suited for. The operating model is authoritative about structure: what exists, what it means, how it depends, who owns it, what is permitted. AIOps is authoritative about nothing — and excellent at something the model is not: reasoning over time and behavior, detecting patterns in high-volume signal, surfacing the unexpected. The failure of the last decade was asking AIOps to supply structural truth it could only infer. The

correction is to let the operating model supply structure and let AIOps do behavioral analysis on top of it.

Read this way, the IOM does not compete with AIOps or replace it. It completes it. The most sophisticated analysis layer in the stack finally gets a reference frame worthy of its sophistication — and the result is the AIOps that was promised, not because the intelligence improved, but because it was finally given something true to be intelligent about.

> ***AIOps was never the problem. Its ground truth was. Give it an authority layer to reason against, and the analysis that always looked promising finally becomes trustworthy.***

10. Conclusion

The case against AIOps was always overstated, and the case for better AIOps algorithms was always going to disappoint, because both mistook the location of the problem. AIOps reasons over telemetry, and telemetry cannot tell it what is intended or permitted. An Infrastructure Operating Model supplies exactly that — an authoritative, continuously reconciled model of intent that AIOps consumes as ground truth while the operating model remains the authority above it. Grounded this way, correlation becomes intent-relative, noise collapses to genuine deviation, and root cause becomes deterministic. The path to AIOps that organizations actually trust does not run through a smarter model. It runs through an operating model beneath it.

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