Preprints.org

Article

Not peer-reviewed version

The Missing Layer in Modern IT: Governance of Commitments, Not Just Compute and Data

Rao Mikkilineni * and W. Patrick Kelly

Posted Date: 5 March 2026

doi: 10.20944/preprints202603.0413.v1

Keywords: coherence debt; governance; CAP theorem; FCAPS; autonomic computing; digital genome; knowledge networks; structural machines; DIME; AMOS; service orchestration

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

The Missing Layer in Modern IT: Governance of Commitments, Not Just Compute and Data

Rao Mikkilineni * and W. Patrick Kelly

Opos Solutions

* Correspondence: rao@opos.ai

Abstract

Contemporary enterprise IT operations are implemented largely atop Shannon–Turing computing: programs execute read–compute–write cycles over data structures, while governance (fault handling, configuration control, auditability, and continuity) is applied externally through infrastructure platforms, observability stacks, and human processes. This separation scales analytic throughput but accumulates coherence debt: locally expedient commitments whose provenance and revisability degrade until exposed by shocks (failures, security incidents, regulatory demands, or architectural transitions). We synthesize a model evolution that integrates computation with regulation at two distinct levels: (i) Distributed Intelligent Managed Elements (DIME), which modifies the Turing cycle to read–check-with-oracle–compute–write by infusing a signaling overlay and FCAPS (Fault, Configuration, Accountability, Performance, Security) supervision into computation in progress; and (ii) AMOS, which fully decouples the process executor from governance by treating any Turing-equivalent engine as a replaceable execution substrate while elevating knowledge structures—encoded as local and global Digital Genomes—to first-class operational state in a governed Knowledge Network. We further present implementation evidence via a microservice transaction testbed that operationalizes dynamic topology as data, a capability-oriented control plane, decoupled application-layer FCAPS from IaaS/PaaS FCAPS, and policy-selectable consistency/availability semantics. We argue that the principal benefit of AMOS is not "circumventing" impossibility results such as CAP, but governing their trade-offs as explicit commitments with auditable lineage and controlled convergence back to coherent state.

Keywords: coherence debt; governance; CAP theorem; FCAPS; autonomic computing; digital genome; knowledge networks; structural machines; DIME; AMOS; service orchestration

1. Introduction

Big data analytics (BDA) and cognitive computing increasingly sit inside core business processes—pricing, credit and fraud decisions, supply-chain actions, personalization, and compliance—where outputs are not merely "predictions" but operational commitments with economic, legal, and reputational consequences. There is a compelling need arising for integrative approaches that connect BDA, AI, and cognitive computing to business model innovation (BMI) and sustainable value creation, beyond isolated technical advances.

Yet most enterprise IT operations remain grounded in an implementation regime shaped by Shannon–Turing structures: computation is realized as read–compute–write transformations over data structures, while governance (fault handling, configuration control, auditability, continuity, and security) is applied externally via IaaS/PaaS controls, observability tooling, and human incident processes [1,2]. This separation scales throughput, but it is structurally weak at reducing operational complexity, because each new layer added to restore reliability (retries, sidecars, controllers, policy scripts, runbooks) increases indirection while leaving the meaning and warrant of commitments under-specified inside the computing model.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

A succinct way to name the limitation is the absence of model closure: general-purpose computers can model external physical systems with arbitrary precision, but face logical constraints when the modeled world includes the computer itself. Cockshott and colleagues capture this in the observation: "We can use them to deterministically model any physical system, of which they are not themselves a part." The operational consequence is practical: when governance remains outside computation, a system cannot natively represent—and therefore cannot reliably govern—the operational context, constraints, and commitments that determine safe, explainable behavior under uncertainty [3,4].

This paper focuses on the resulting organizational liability: coherence debt. We define coherence debt as the accumulated liability created when operational commitments (decisions encoded in configurations, workflows, policies, model deployments, routing rules, and automated actions) cannot be reconstructed, criticized, and safely revised because their justificatory lineage has degraded [5]. Coherence debt increases when the system fails to preserve (i) provenance (inputs, context, dependencies, versions), (ii) warrant (policy constraints and reasons that made an action admissible), and (iii) repair obligations (the required convergence pathway back to intended invariants under partial failure). Coherence debt is distinct from technical debt [6]: it is not primarily a code-quality problem, but a governance-of-commitments problem that manifests as brittle incident response, opaque AI behavior, costly change failure, and escalating integration complexity.

The problem becomes acute in distributed big-data systems where partitions and partial failures are normal. Impossibility results such as CAP do not "go away"; rather, under Shannon–Turing SOTA they are frequently handled implicitly through timeouts, retries, and emergent platform behavior, producing accidental divergence and post-hoc reconciliation [7]. As BDA and cognitive computing are embedded deeper into decision workflows, this implicitness becomes a barrier to BMI: organizations cannot sustain explainability, accountability, and regulatory readiness if the governance of commitments is not first-class.

Motivated by this gap, we present a coherent evolution of operations models that integrates computation and regulation at two distinct levels:

Shannon–Turing SOTA operations, where computation and governance are separated and CAP trade-offs often emerge as incident-time surprises.

DIME (Distributed Intelligent Managed Elements), which modifies execution semantics toward read–check-with-oracle–compute–write by infusing a signaling overlay (addressing, alerting, mediation, supervision) closer to computation-in-progress, thereby reducing accidental divergence.

AMOS, which fully decouples the process executor from governance and elevates knowledge structures—encoded as local and global Digital Genomes—into a governed Knowledge Network where workflow regulation is implemented as explicit constraints (constitution, legislation, self-regulation of form and function).

Our central claim is deliberately conservative: these models do not violate distributed impossibility results; rather, they operationalize trade-offs as explicit, auditable commitments with defined degradation and convergence pathways, thereby reducing coherence debt while improving the reliability and governability required for AI-enabled business model innovation [8].

2. Background and Model Evolution: SOTA → DIME → Mindful Machines (AMOS)

In this section, we will review the evolution of computing structures and their impact on Coherence Debt in IT Operations.

2.1. Shannon–Turing SOTA IT Operations: Architecture and Structural Shortfalls

Modern "state-of-the-art" enterprise IT operations are largely built on the computing model shaped by Claude Shannon and Alan Turing: read–compute–write over data structures, with reliability and governance provided by external control layers. In practice, this is realized through

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

cloud-native stacks that separate a control plane (desired state, scheduling, policy distribution) from a data plane (workload execution and traffic). For example, Kubernetes formalizes cluster control-plane controllers and node-level agents to place and maintain workloads, providing "self-healing" via reconciliation loops [9]. Likewise, service meshes such as Istio introduce sidecar proxies in the data plane and a control plane that programs routing and security policies [10].

Shortcoming (Relevant to Complexity and Coherence Debt)

These architectures excel at scaling compute and managing infrastructure churn, but they externalize the governance of meaningful commitments—"what must remain true," "what counts as an acceptable degraded mode," "what evidence justified an automated action," and "how we converge back to coherence." As a result:

- Governance becomes a patchwork of infra policies, mesh rules, retries/timeouts, and human runbooks—often non-composable across layers.
- Failure handling is frequently implicit (emergent from timeouts, retries, backoff, and partial writes), so the enterprise discovers its effective semantics during incidents rather than selecting them as auditable commitments.
- The system manages "compute and connectivity," but not the computer and the computed (the governed knowledge/commitment structures that must remain stable under uncertainty), which drives operational opacity and escalating integration complexity.

This is the architectural pathway by which Shannon–Turing SOTA stacks tend to increase coherence debt as organizations scale BDA and AI into decision workflows.

2.2. DIME Computing: Historical Response to SOTA Shortfalls

The Distributed Intelligent Managed Element (DIME) computing model emerged as an explicit attempt to address the "externalized governance" problem by infusing FCAPS (fault, configuration, accounting, performance, security) management into the computing substrate via a signaling overlay and supervision mechanisms. In DIME, each node remains a conventional executor (a Turing-equivalent process), but it is paired with management and signaling structures so the distributed workflow can be governed "in parallel" with computation.

A canonical expression of this idea is to modify the basic loop from read–compute–write to a regulated form that resembles read–check(with oracle)–compute–write, inspired by Alan Turing's oracle-machine framing: computation proceeds, but is gated or supervised by context-sensitive management channels and best-practice "oracle" guidance [11,12].

2.2.1. What DIME Solved (Relative to SOTA)

DIME's key innovation is architectural: it reduces reliance on ad-hoc external management layers (hypervisors, after-the-fact scripts, manual interventions) by making supervision and signaling part of the distributed execution fabric. This supports patterns such as auto-failover, auto-scaling, live migration, and workflow resilience with less external glue code.

2.2.2. Why DIME Is Still Incomplete for "AI-Era Governance."

DIME substantially improves managed execution, but it largely remains a managed-process paradigm: it strengthens FCAPS and supervision around executors, yet does not fully elevate knowledge/meaning (semantic commitments, warrant, provenance, revision rules) into first-class operational structures shared across the system. This becomes limiting when enterprises deploy BDA/AI inside workflows that must be explainable, auditable, and revisable as part of business model innovation (BMI) [13–17]. The subsequent evolution toward Mindful Machines can be understood as closing this gap: moving from "managed computation" to "governed knowledge-centric commitments."

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

(That interpretation is supported by later Mindful Machines work explicitly positioning itself as a shift from conventional and agentic AI toward integrated governance, traceability, and self-regulation grounded in knowledge structures.)

2.3. From Data Structures to Knowledge Structures: GTI, BMT, PMK, and Deutsch

Mindful Machines extend the DIME motivation (embed regulation) but shift the object of regulation from processes and infrastructure toward knowledge-bearing structures that govern commitments under uncertainty.

2.3.1. General Theory of Information and Operational Schemas

Mark Burgin's General Theory of Information (GTI) provides a broader basis for treating information/knowledge as more than signal or data [18]. In the Mindful Machines lineage, GTI motivates representing enterprise operations as schemas/structures that can be reasoned about and governed—not merely as data flowing through compute pipelines.

A concrete step in this direction is the transition "from data processing to knowledge processing" via operational schemas—structures encoding roles, relations, constraints, and allowable transformations as manipulable operational artifacts [19,20].

2.3.2. Burgin–Mikkilineni Thesis (BMT) and Structural Machines

The Burgin–Mikkilineni Thesis (BMT) frames autopoietic and cognitive behavior in artificial systems as requiring multi-level information processing and, axiomatically, efficient realization via structural machines (machines operating on structures, not only symbol strings) [21,22]. This provides the theoretical bridge from "managed executors" to "knowledge-centric substrates," consistent with using graphs/networks as first-class computing objects rather than incidental storage.

2.3.4. Deutsch: Discernible Explanations as the Unit of Knowledge

David Deutsch's epistemic framework emphasizes explanatory knowledge—good explanations that are hard to vary and that enable reliable intervention in reality [23]. In this paper's framing, Deutsch provides the "why" behind coherence debt: when systems cannot preserve the explanatory lineage of commitments (what was believed, why it was believed, what constraints applied, how it can be revised), organizations lose the ability to reliably shape outcomes—precisely the failure mode seen when AI outputs are operationalized without governance.

2.4. AMOS: Practical Implementation of Mindful Machines

AMOS is presented as an implementation substrate that decouples executors from governance. Instead of modifying each executor to embed the oracle/FCAPS manager (DIME's direction), AMOS treats any Turing-equivalent component as a replaceable process execution engine and elevates governance into shared knowledge structures (Digital Genomes) within a governed knowledge network [24–26].

Implementation evidence reported in Mindful Machines work includes:

- A Digital Genome approach for specifying distributed application structure and behavior, with associative memory and event-driven history for traceability and adaptive regulation (e.g., video streaming/service continuity) [27].
- An AMOS-orchestrated loan default prediction system demonstrating autopoietic regulation (deployment, scaling, recovery), cognitive network management (connectivity semantics), and workflow preservation under disruption—explicitly arguing a shift from static prediction pipelines to governed, knowledge-centric systems [28].
- Formal grounding that explicitly integrates GTI, BMT, and Deutsch's epistemic stance in the architecture narrative for Mindful Machines [29].

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

Net evolution claim (Section 2 synthesis):.

- SOTA (Shannon–Turing): strong execution scale; governance externalized; coherence debt rises as layers accumulate.
- DIME: governance infused nearer to execution via oracle-like supervision and FCAPS overlays; reduces ad-hoc external management, but remains centered on managed processes.
- Mindful Machines (AMOS): governance becomes first-class knowledge structure (Digital Genomes + event histories + policy constraints), with executors decoupled and coordinated through a governed knowledge network.

2.5. Bridge to Section 3: AMOS, CAP Trade-Offs, and Decoupling "Computer vs. Computed"

Section 3 builds on this evolution to analyze distributed database behavior under partition: CAP does not disappear, but AMOS reframes CAP as explicit, governable commitments. The key move is the decoupling of the computer (process executors, storage engines) from the computed (knowledge structures encoding policy, warrant, topology, and convergence obligations). In this framing, CAP trade-offs become selectable and auditable at the workflow/knowledge-governance layer, rather than being discovered incident-by-incident as emergent behavior of timeouts, retries, and infrastructure controllers.

3. AMOS and the Governance of CAP Trade-Offs: Decoupling the Computer from the Computed

The CAP theorem formalizes a constraint: in a partitioned distributed system, designers cannot simultaneously guarantee consistency and availability. In practice, CAP is not merely a database property; it is a constraint on enterprise commitments (transactions, decisions, policy-gated actions) executed across distributed components under partial failure. When those commitments are made implicitly—through timeouts, retries, default replication behavior, and ad-hoc failover scripts—organizations accumulate coherence debt because they cannot reliably explain what happened, why it was allowed, and how the system is obligated to converge back to a coherent state.

This section argues that the decisive innovation in AMOS is to treat CAP trade-offs as governable commitments encoded in knowledge structures, not emergent properties of "whatever the platform did during the outage."

3.1. SOTA: CAP Trade-Offs as Emergent Behavior and Incident-Time Surprises

Shannon–Turing SOTA operations typically handle CAP implicitly. Even when teams intend a "CP" or "AP" posture, the effective posture is often defined by:

- client retry patterns,
- load balancers and service mesh timeouts,
- leader election and failover defaults,
- cache staleness and read fallback,
- asynchronous replication lag,
- and human incident actions.

The result is a familiar operational anti-pattern: the system's actual semantics are discovered during failures, and then retrofitted into runbooks and postmortems. That is coherence debt accumulation by construction.

3.2. DIME: Oracle-Mediated Gating Close to Execution

The DIME computing model arose as a response to this shortfall by embedding management (FCAPS) into the execution fabric using a signaling overlay and supervised process dynamics. DIME prototypes are explicitly described as incorporating FCAPS via a signaling network overlay and dynamic control of DIME nodes.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

Framed as execution semantics, DIME pushes systems toward a regulated loop:

- read → check (oracle/supervision) → compute → write

Operational implication for CAP: DIME does not invalidate CAP, but it reduces accidental divergence by allowing the "check/supervision" phase to:

- block unsafe mutations during suspected partition conditions,
- force an explicit degraded mode,
- redirect computation to quorum-reachable components,
- and accelerate recovery actions via supervision that is closer to runtime execution than post-hoc ops tooling.

DIME is therefore best understood as making CAP trade-offs explicitly mediated near the compute step, rather than left to emergent failure behavior.

3.3. AMOS: CAP as a Governed Commitment in a Knowledge Network

AMOS extends the DIME motivation (integrate regulation) but changes where and what is regulated.

- DIME integrates regulation into the executor's runtime loop.
- AMOS decouples the executor and makes the regulated object a knowledge structure: policies, topology, invariants, provenance, and convergence obligations encoded in Digital Genomes and enforced through workflow regulation.

This approach is articulated in the Mindful Machines implementation literature by Rao Mikkilineni and W. Patrick Kelly (and coauthors such as Gideon Crawley) as a shift from treating knowledge as "after-the-fact annotation" to treating it as an organizing substrate that shapes computation

3.3.1. Decoupling "Computer" vs "Computed"

In AMOS, the computer is any process executor (symbolic, sub-symbolic, probabilistic, LLM, rules engine, database engine, human). The computed is not merely data output; it is the governed commitment structure:

- what invariants are in force,
- what evidence/policy warranted an action,
- what degradation is permitted,
- and what convergence workflow is mandatory after disruption.

AMOS integrates computer and computed by making the computed commitments explicit, versioned, and enforceable in a shared governance substrate (local + global Digital Genome), rather than implicit in emergent platform behavior.

3.3.2. CAP in AMOS: From "Impossibility" to "Policy-Governed Trade Space"

Under AMOS, CAP is handled as follows:

- Partition tolerance is assumed (not treated as an exception).
- Consistency vs availability becomes a workflow policy choice, not an accident:
  - some workflows are CP-leaning (block or degrade under partition),
  - some are AP-leaning (continue with bounded divergence),
  - many are "governed hybrid" (quorum rules, bounded staleness, compensation).
- Divergence is treated as governed coherence debt, meaning:
  - it must be recorded with provenance,
  - bounded by explicit rules,
  - and reconciled by a defined convergence workflow.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

This is the precise, defensible claim: AMOS doesn't "circumvent CAP"; it operationalizes CAP trade-offs as auditable commitments.

3.4. Concrete Mechanisms: How AMOS Implements Governed CAP Trade-Offs (with Evidence)

Topology-as-Data: Governance Starts by Making Structure Explicit

AMOS implements governed CAP trade-offs by making the conditions of distributed commitment explicit and machine-actionable, rather than letting them emerge from timeouts and retries. In the Transactions/AEM+proxy testbed, this begins with topology-as-data:

- services ingest runtime connectivity payloads via /network-config, using typed edges (e.g., API↔AEM, AEM↔proxy) and
- an idempotent guard so the system's dependency graph is first-class operational state, not hidden in DNS or deployment YAML.

This matters for CAP because, under partitions, reachability and quorum eligibility must be computed from governed structure, not guessed from incidental failures. On top of this explicit structure, the AEM encodes the CAP posture as an explicit commitment policy—all, quorum, or one—that determines how multi-backend operations are adjudicated:

- all is CP-leaning (commit only if every target succeeds),
- quorum is a governed hybrid (bounded availability with majority agreement), and
- one is AP-leaning (prioritize availability while explicitly accepting divergence and therefore creating a stronger reconciliation obligation).

The key difference from SOTA is that these trade-offs are not implicit in client retry logic; they are inspectable, auditable workflow rules attached to execution [26].

In summary:

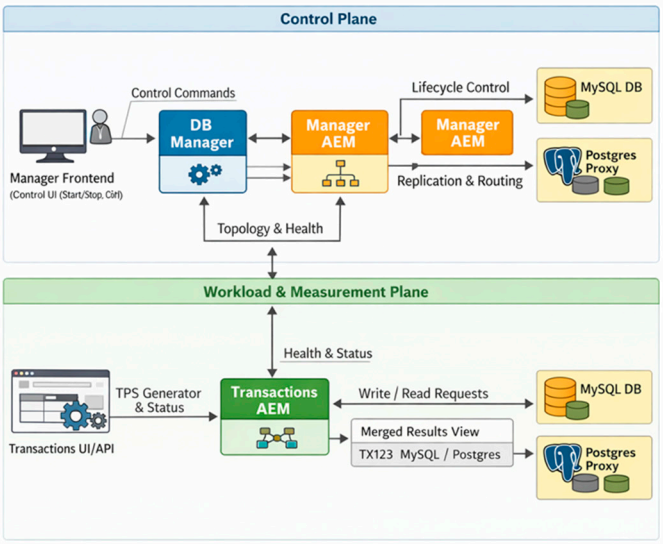
1. SOTA tends to realize CAP trade-offs as emergent behavior; coherence debt accumulates because commitments are not first-class.
2. DIME improves matters by introducing supervised, oracle-mediated gating closer to execution via a signaling overlay.
3. AMOS/Mindful Machines go further by decoupling executors and elevating governance into knowledge structures (Digital Genomes + policy-regulated workflows), making CAP trade-offs explicit, auditable commitments with defined degraded modes and convergence pathways.

4. Architectural Design and Implementation of a Capability-Oriented Distributed Orchestration Framework

The system represents a fundamental shift from traditional database-centric architectures toward a capability-oriented control plane. Legacy designs often require the orchestration logic to possess direct knowledge of database types, drivers, and schemas, which tightly couples logic to storage details and hinders scalability. By introducing generic Proxy Services, the architecture achieves a clean separation of concerns: the Application Event Manager (AEM) manages coordination and policy, while the proxies encapsulate the complexities of specific backends. This allows the system to evolve and extend through deployment rather than core code modifications, mirroring modern service meshes and data fabrics.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1



The Control and Data Planes

The framework is bifurcated into a Control Plane and a Workload & Measurement Plane, implemented as a containerized microservice testbed.

4.1. Database Proxy Services (MySQL and PostgreSQL)

The proxies provide a uniform HTTP control surface, abstracting backend-specific operations.

- Lifecycle Management: Proxies support multiple control drivers, including Docker, system control, and no-op modes.
- State Tracking: They monitor granular lifecycle states, including running, stopped, starting, stopping, and degraded.
- Health and Schema: Proxies execute periodic SELECT 1 queries to verify reachability and support the auto-creation of transaction tables at startup.

4.2. Application Event Manager (AEM)

The AEM acts as a stateless fanout broker responsible for routing commands and maintaining target registries.

- Execution Strategies: It supports diverse routing policies, including broadcast (all/quorum/one), round-robin, and primary-secondary failover.
- Replication Coordination: The AEM facilitates synchronization between heterogeneous engines through replication watermark tracking and incremental row transfers.

4.3. Database Manager (DB_Manager) and UI

The DB_Manager serves as the primary control plane, coordinating infrastructure and issuing lifecycle commands. It communicates with a web-based frontend that visualizes system state and provides interactive controls. To ensure robustness, the manager can bypass the AEM and invoke proxies directly if communication fails.

4.4. Dynamic Topology and Service Discovery

To avoid the brittleness of static addressing, the system utilizes Structural Event Manager (SEM) payloads for runtime configuration.

- Graph-Based Discovery: Services receive a topology payload describing connectivity via typed edges (e.g., API_AEM, AEM_MYSQL).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

- Idempotency and Safety: Configuration updates use a config_id guard to prevent repeated application. State management employs thread-safe locks to protect registries and lifecycle status during concurrent updates.

Table 1. Features and Implementation details.

Feature	Implementation Detail
ID Allocation	Uses a locked, in-memory allocator (NEXT_ID) seeded from the AEM's maximum ID to prevent collisions during bursts.
Fault Isolation	Wraps upstream failures as HTTP 502 errors to distinguish backend issues from local API faults.
Data Merging	The UI merges records by ID, displaying MySQL and PostgreSQL observations in a single row for consistency analysis.
Availability Signals	Displays "not available" based on AEM state to prevent misinterpretation of missing records.

Appendix A provides the descition of a Video and a link that demonstrate the implementation.

5. Lessons Learned

This section distills the engineering and conceptual lessons from implementing and exercising the Transactions testbed, with emphasis on how the design reduces coherence debt and clarifies CAP trade-offs in practice.

5.1. Make Topology a First-Class Operational Artifact or You Cannot Govern Distributed Behavior

The most consequential practical shift was treating service connectivity as explicit state rather than an implicit byproduct of deployment tooling. Implementing /network-config across services and applying topology payloads idempotently via config_id converted "what depends on what" into inspectable, replayable knowledge.

Lesson: If reachability and dependency are not first-class, CAP behavior becomes an emergent accident of timeouts and retries, and incident explanations remain narrative rather than reconstructable.

5.2. The Capability/Proxy Boundary Dramatically Lowers Integration Complexity

Encapsulating each database behind a uniform proxy interface eliminated database-specific handling from the orchestration layer, enabling extensibility by deployment (add a proxy) instead of orchestration code changes.

Lesson: This boundary is a practical mechanism for transitioning from data-structure coupling (clients embed storage semantics) to knowledge-structure coupling (clients reason over capabilities and policy).

5.3. Explicit Commitment Policies (All/Quorum/One) Turn CAP from Incident Behavior into Design Behavior

Implementing requirement policies in AEM made the "consistency vs availability" posture explicit and inspectable rather than an implicit outcome of infrastructure defaults.

Lesson: Most organizations already operate with different semantics across workflows (e.g., payments vs recommendations). Encoding these differences as explicit policies is the first step toward governing CAP trade-offs as commitments rather than discovering them in outages.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

5.4. Health Checks Are Necessary but Not Sufficient—Systems Need Declared Degraded States and Admissibility Rules

The system benefited from health probing and routing strategies, but the more important lesson was that "up" and "down" are too coarse. Proxies explicitly tracked lifecycle states including degraded, and surfaced them through health endpoints.

Lesson: To reduce coherence debt, a system must encode admissibility: which operations are permitted under which states, and what obligations are created (e.g., reconciliation after AP-leaning operation).

5.5. Decoupling Application FCAPS from IaaS/PaaS FCAPS Enables Portability and Auditability

By exposing lifecycle controls through multiple actuation drivers (Docker/systemcontrol/no-op), the proxies supported both "self-managed" and "externally managed" databases without changing higher-level policy logic.

Lesson: Substrate self-healing (restart/reschedule) does not address semantic correctness. Application FCAPS must exist above the substrate to govern commitments, especially when BDA/AI workflows are embedded in operations.

5.6. A Stateless Coordinator Is a Strength When Coupled to an Explicit Knowledge Substrate

The AEM's statelessness simplified reliability and scaling while still allowing rich behavior via explicit configuration (targets + policies).

Lesson: "Intelligence" should not be hidden in mutable coordinator state; it should live in governed knowledge objects (topology, policy, provenance), which can be versioned, audited, and replayed.

5.7. Fallback Pathways Reduce Single-Point Brittleness and Improve Explainability During Incident Response

Allowing DB_Manager to call proxies directly when AEM is unavailable improved robustness and reduced "control plane outage cascades."

Lesson: Incident response often fails because operators cannot act when the control plane is degraded. Designing explicit fallback control paths reduces coherence debt by making "how to act during failure" part of the system, not just a runbook.

5.8. Convergence Must Be Treated as a First-Class Workflow, Not a Background Hope

The testbed's replication mechanisms (watermarks, incremental transfer) highlight the core requirement for any AP-leaning behavior: divergence is acceptable only if convergence is governed.

Lesson: In coherence-debt terms, divergence creates an obligation. If the obligation is not structurally encoded, the organization pays later through manual reconciliation, data disputes, and compliance risk.

5.9. UI Design Is Part of Governance: "Not Available" Is Better than Silent Absence

The UI explicitly displayed backend "not available" states to prevent misinterpretation of missing records as semantic absence.

Lesson: Coherence debt often grows through interpretive errors. Operator-facing surfaces should encode uncertainty and partial failure explicitly to preserve correct reasoning.

5.10. Implication for AMOS/Mindful Machines: The Testbed Identifies the Minimum Viable "Governance Substrate"

Across experiments, the irreducible set of primitives for coherence-debt reduction were:

- Topology-as-data with idempotent updates.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

- Capability boundaries (proxies) separating orchestration from backend specifics.
- Explicit commitment policies (all/quorum/one; routing strategies).
- Declared degraded modes + lifecycle state for admissibility.
- Convergence workflows (replication/reconciliation) as obligations.

Lesson: These primitives are sufficient to demonstrate how an AMOS-style approach can govern CAP trade-offs as auditable commitments and reduce coherence debt without requiring immediate replacement of existing executors (databases, services, or ML components).

6. Conclusions

This paper sets out to articulate IT professionals and researchers on an alternative way to improve IT operations while decreasing coherence debt—especially in enterprises where big data analytics and cognitive computing are embedded in mission-critical decision workflows. Aligning with the integrative perspectives connecting BDA, cognitive computing, and business model innovation, we framed operational reliability not only as an infrastructure problem, but as a governance-of-commitments problem.

We made three core contributions:

1. A coherent model evolution: We contrasted (i) Shannon–Turing SOTA operations—where computation and governance are separated and complexity is managed externally—against (ii) DIME, which reduces accidental divergence by infusing oracle-like supervision and signaling closer to computation, and (iii) AMOS/Mindful Machines, which decouple process executors and elevate governance into knowledge structures (Digital Genomes) within a governed knowledge network.
2. A conservative, technically defensible CAP claim: We emphasized that CAP is not "circumvented" but can be governed. AMOS enables explicit, auditable selection of consistency/availability trade-offs as workflow commitments, with defined degraded modes and convergence obligations rather than incident-time surprises.
3. Implementation evidence and migration primitives: Using the Transactions testbed, we demonstrated the minimum operational primitives required to reduce coherence debt: topology-as-data with idempotent updates, capability boundaries via database proxies, explicit commitment policies (all/quorum/one), declared degraded states and lifecycle admissibility, and reconciliation mechanisms as first-class convergence workflows.

Taken together, these results support the main thesis: coherence debt is the hidden limiter of AI-enabled business transformation, and it grows when governance remains external to computation. DIME reduces coherence debt by moving supervision closer to the execution loop; AMOS reduces coherence debt more fundamentally by making governance a first-class knowledge substrate that decouples the "computer" (executors) from the "computed" (governed commitments, warrant, and convergence obligations).

6.1. Future Work

Future work is needed across theory, engineering, and empirical evaluation to advance coherence-debt–aware operations from prototype demonstrations to widely adoptable enterprise practice.

6.1.1. Formalizing Coherence Debt Metrics and Benchmarking Protocols

While this paper operationalized coherence debt using practical proxies (decision reconstruction time, provenance completeness, recovery energy), a more rigorous benchmarking suite should be developed, including:

- standardized failure injection (network partitions, replica loss, partial writes),
- workload classes (strict consistency vs bounded divergence vs read-only degrade),

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

- and governance maturity measures (policy versioning, audit completeness, reconciliation determinism).

A useful outcome would be a repeatable benchmark analogous to fault-tolerance test harnesses, but focused on governance of commitments rather than only throughput or availability.

6.1.2. Extending Digital Genome Expressivity for Workflow Commitments

The testbed demonstrates policy knobs (requirement policies, routing strategies) and runtime topology. A next step is to encode a richer Digital Genome schema for:
invariants and admissibility constraints (what must never happen),

- allowable degraded modes per workflow,
- reconciliation obligations (who/what/when/how to converge),
- and explicit provenance fields for warrant and policy lineage.

This would strengthen the "constitution/legislation/self-regulation" layer from conceptual framing into machine-interpretable governance.

6.1.3. Integrating AI/Cognitive Components as Governed Executors

A critical direction for BDCC relevance is to integrate cognitive computing executors (e.g., retrieval-augmented LLM services, anomaly detectors, forecasting models) as nodes in the same governed knowledge network. The key research question is: what additional governance is required for AI-driven commitments (e.g., confidence thresholds, counterfactual checks, explanation constraints, and human-in-the-loop escalation policies) to reduce coherence debt rather than amplify it?

6.2.4. Multi-Region and Multi-Cloud Experiments with Explicit CAP Policy Selection

The Transactions testbed can be extended to:
multi-region deployments with controlled partition scenarios,
heterogeneous latencies and asymmetric partitions,
and different storage replication models.
The goal is to quantify how explicit policy selection (all/quorum/one) and governed convergence workflows change failure outcomes, reconciliation cost, and decision auditability compared to SOTA defaults.

6.1.5. Security and Compliance Governance as First-Class Commitments

The present implementation focuses primarily on fault/config/performance semantics. Future work should add explicit identity, authorization, and compliance policy enforcement—particularly:

- signed topology payloads and policy objects,
- provenance integrity guarantees,
- and audit logs suitable for regulatory workflows.

This is essential for demonstrating that coherence debt reductions translate into compliance readiness—directly relevant to BMI and sustainable AI adoption.

6.1.6. Longitudinal Field Studies Linking Governance Maturity to Business Model Outcomes

Finally, to connect architecture choices should address:

- lower incident cost variance,
- lower change failure rate,
- faster time-to-certify AI-enabled workflows,
- and improved trust/retention outcomes attributable to explainable, auditable operations.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

6.3. Closing Remark

The long-term implication is that enterprise advantage shifts from "more analytics" to better governed commitments. In that sense, AMOS/Mindful Machines propose not merely an operations toolkit but an evolution of the computing model: a move from externally governed Shannon–Turing execution toward knowledge-centric, self-regulating structural machines that can sustain trustworthy BDA and cognitive computing at scale. Appendix B provides a comparison between the current state of the art and AMOS governance.

Supplementary Materials: The following supporting information can be downloaded at: Preprints.org, Video S1: AMOS Governed Distributed Databases and CAP Regulation (https://youtu.be/RW0VjpWbwu4).

Author Contributions: "Conceptualization, R.M.; methodology, R.M.; software, P.K.; validation, R.M., and P.K.; formal analysis, R.M.; investigation, P.K.; writing—original draft preparation, R.M.; writing—review and editing, R.M.; visualization, P.K.; All authors have read and agreed to the published version of the manuscript."

Funding: "This research received no external funding"

Institutional Review Board Statement: "Not applicable"

Informed Consent Statement: "Not applicable."

Data Availability Statement: "Not Applicable"

Acknowledgments: ChatGPT , GEMINI, and CoPILOT. The authors have reviewed and edited the output and take full responsibility for the content of this publication."

Conflicts of Interest: "The authors declare no conflicts of interest."

Abbreviations

The following abbreviations are used in this manuscript:

GTI	General Theory of Information
BMT	Burgin-Mikkilineni Thesis
FCAPS	Fault, Configuration, Accounting, Performance, and Security letter acronym
SOTA	State Of The Art
BDA	Big Data Analytics
BMI	Business Model Innovation
CAP	Consistency, Availability, Partitioning
AMOS	Autopoietic and Metacognitive Operating System

Appendix A. Description of the Video

The video [30] is a practical demo of AMOS as a governance layer for distributed database operations. It shows how AMOS sits above the application, database manager, and database proxies to make runtime behavior visible and controllable. Instead of relying on fixed failover rules hidden in infrastructure, the system exposes topology, health, routing strategy, and transaction state in real time. The demo walks through normal operation, a partial failure scenario, continued service during degradation, and recovery after the failed component returns—showing that AMOS can keep the application running while making backend divergence and re-synchronization explicit.

Key points made:

- AMOS is a control plane, not just a monitoring dashboard.
- It separates control/governance from transaction workload and measurement.
- Topology and health are explicit runtime knowledge and visible to operators.
- Routing and failover behavior can be changed live through policy.
- The system can continue operating during partial failures.
- Backend divergence is made visible, rather than hidden.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

- Recovery includes re-alignment and re-synchronization after the failed path returns.

The approach emphasizes policy-driven resilience, observability, and operational control for IT.

Appendix B. Comparison Between Current State of the Art and AMOS Approach

Table 2. Comparing Current IT State of the ART and AMOS Approach.

CURRENT STATE OF THE ART	AMOS GOVERNANCE
Resilience is pattern-driven: reliability is achieved through retries, circuit breakers, health checks, redundancy, failover, and graceful degradation implemented across app code, platform settings, and infrastructure policies.	**Resilience is governance-driven**: runtime behavior is steered through an explicit control layer that applies policy over topology, routing, and backend participation.
Control is fragmented: service mesh, orchestrator, database config, load balancer, and application logic each manage a piece of failure handling.	**Control is unified**: routing mode, failover behavior, proxy status, and recovery actions are coordinated from one control surface.
Control is fragmented: service mesh, orchestrator, database config, load balancer, and application logic each manage a piece of failure handling.	**Control is unified**: routing mode, failover behavior, proxy status, and recovery actions are coordinated from one control surface.
Failover behavior is usually preconfigured: teams define static policies ahead of time and depend on infrastructure to execute them.	**Failover behavior is runtime-selectable**: policy can change live based on current health, topology, or business intent.
Partial failure is handled by masking or routing around faults: the system aims to preserve availability, often without showing application-level divergence clearly.	**Partial failure is handled explicitly**: the system continues service, exposes divergence between backends, and governs recovery back to alignment.
Consistency/availability trade-offs are implicit in platform design choices: teams choose database modes, replicas, and failover patterns up front.	**Consistency/availability trade-offs are explicit policy choices**: commitment posture can be governed as an operational decision rather than buried in infrastructure defaults.
Operations are largely reactive: telemetry informs human operators or automation after symptoms appear.	**Operations are more anticipatory and controlled**: governance logic can directly shape how the system degrades and recovers.
Best fit: mature cloud-native teams optimizing within existing platform primitives.	**Best fit**: enterprises that want stronger runtime control, explicit policy, and app-visible governance of distributed commitments.

References

1. C. E. Shannon, "A mathematical theory of communication," in The Bell System Technical Journal, vol. 27, no. 3, pp. 379-423, July 1948, doi: 10.1002/j.1538-7305.1948.tb01338.x.
2. Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, Series 2, 42(1), 230-265.
3. Cockshott, Paul & Mackenzie, Lewis & Michaelson, Greg. (2012). Computation and its Limits.
4. Gödel, K., Meltzer, B., & Schlegel, R. (1966). On Formally Undecidable Propositions of Principia Mathematica and Related Systems.
5. Brandom, R. B. (1994). Making it explicit: Reasoning, representing, and discursive commitment. Harvard University Press.
6. P. Kruchten, R. L. Nord and I. Ozkaya, "Technical Debt: From Metaphor to Theory and Practice," in IEEE Software, vol. 29, no. 6, pp. 18-21, Nov.-Dec. 2012, doi: 10.1109/MS.2012.167.
7. Seth Gilbert and Nancy Lynch. 2002. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News 33, 2 (June 2002), 51–59. https://doi.org/10.1145/564585.564601
8. Mikkilineni, Rao. (2012). Distributed Intelligent Managed Element (DIME) Network Architecture Implementing a Non-von Neumann Computing Model. 10.1007/978-1-4614-1924-2_3.
9. Burns, B.; Grant, B.; Oppenheimer, D.; Brewer, E.; Wilkes, J. Borg, Omega, and Kubernetes. Queue 2016, 14, 70–93, https://doi.org/10.1145/2898442.2898444.
10. Sharma, Rahul & Singh, Avinash. (2020). Istio VirtualService. 10.1007/978-1-4842-5458-5_4.
11. Mikkilineni, R. (2025). Truth as Survival Architecture: Coherence, Knowledge and Mindful Machines. Preprints. https://doi.org/10.20944/preprints202512.0248.v1

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2026 doi:10.20944/preprints202603.0413.v1

12. A. M. Turing, Systems of Logic Based on Ordinals, Proceedings of the London Mathematical Society, Volume s2-45, Issue 1, 1939, Pages 161–228, https://doi.org/10.1112/plms/s2-45.1.161
13. Burgin, M. Theory of Information: Fundamentality, Diversity, and Unification; World Scientific: Singapore, 2010. [Google Scholar]
14. Burgin, M. Theory of Knowledge: Structures and Processes; World Scientific Books: Singapore, 2016. [Google Scholar]
15. Burgin, M. Structural Reality; Nova Science Publishers: New York, NY, USA, 2012. [Google Scholar]
16. Burgin, M. Triadic Automata and Machines as Information Transformers. Information 2020, 11, 102. [Google Scholar] [CrossRef]
17. Burgin, M.; Mikkilineni, R. On the Autopoietic and Cognitive Behavior. EasyChair Preprint No. 6261, Version 2. 2021. Available online: https://easychair.org/publications/preprint/tkjk
18. Burgin, M. Theory of Information: Fundamentality, Diversity, and Unification; World Scientific: Singapore, 2010.
19. Burgin, M.; Mikkilineni, R. From Data Processing to Knowledge Processing: Working with Operational Schemas by Autopoietic Machines. Big Data Cogn. Comput. 2021, 5, 13. [Google Scholar] [CrossRef]
20. F.G. Varela, H.R. Maturana, R. Uribe, Autopoiesis: The organization of living systems, its characterization and a model, Biosystems, Volume 5, Issue 4, 1974, Pages 187-196, ISSN 0303-2647, https://doi.org/10.1016/0303-2647(74)90031-8 .
21. Maturana, H. R., & Varela, F. J. (1980). Autopoiesis and cognition: The realization of the living. D. Reidel Publishing Company.
22. Deutsch, D. (2011). The beginning of infinity: Explanations that transform the world (1st American ed.). Viking
23. Mikkilineni, R., & Michaels, M. (2025). Physics of Mindful Knowledge. Preprints. https://doi.org/10.20944/preprints202510.1913.v3
24. Mikkilineni, R. (2022). A New Class of Autopoietic and Cognitive Machines. Information, 13(1), 24. https://doi.org/10.3390/info13010024
25. Mikkilineni, R. (2025). General Theory of Information and Mindful Machines. Proceedings, 126(1), 3. https://doi.org/10.3390/proceedings2025126003
26. Mikkilineni, R., Kelly, W. P., & Crawley, G. (2024). Digital Genome and Self-Regulating Distributed Software Applications with Associative Memory and Event-Driven History. Computers, 13(9), 220. https://doi.org/10.3390/computers13090220
27. Mikkilineni, R., & Kelly, W. P. (2025). From static prediction to mindful machines: A paradigm shift in distributed AI systems. Computers, 14, 541. https://doi.org/10.3390/computers14120541
28. Kelly, W. Patrick, Francesco Coccaro, and Rao Mikkilineni. 2023. "General Theory of Information, Digital Genome, Large Language Models, and Medical Knowledge-Driven Digital Assistant" Computer Sciences & Mathematics Forum 8, no. 1: 70. https://doi.org/10.3390/cmsf2023008070
29. Mikkilineni and Kelly, “Medical Knowledge-based Digital Assistant Presentation at HAI2023” https://youtu.be/H0_il7fMl8o (Accessed on 02/15/2026).
30. Mikkilineni and Kelly, “AMOS Governed Distributed Databases and CAP Regulation” https://youtu.be/RW0VjpWbwu4 (Accessed on 03/03/2026)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.