IPHO-Journal of Advance Research in Science And Engineering.
(e-ISSN 3050-8797, p-ISSN 3050-9270)
Volume 01 | Issue12 | December 2023
Published by: International Publication House Open (IPHO)
DOI: https://doi.org/10.5281/zenodo.22278468
Available online at: https://iphopen.org/
Journal homepage: https://iphopen.org/index.php/se

A FULL-STACK PRODUCTION-PLATFORM REFERENCE ARCHITECTURE FOR GOVERNED AGENTIC AND AUTOMATED-ACTION ENTERPRISE SYSTEMS

LAYERED PLATFORM, ORCHESTRATION AND ACTION, AN ANALYTICS-AND-OBSERVABILITY SPINE, CLOSED-LOOP LEARNING, AND OPERATIONAL READINESS - WITH CONFORMANCE PROFILES, ACROSS LLM AGENTS, PROCESS AUTOMATION, INDUSTRIAL CONTROL, DECISION SERVICES, AND INTEGRATED ANALYTICS

MESBAUL HAQUE SAZU[*]
[*]Independent Researcher

ABSTRACT

A large fraction of enterprise AI systems that work in a pilot never operate reliably in production, because a running system must do far more than score a model: it must ingest heterogeneous data, turn it into analytics-driven decisions, execute actions, observe its own behavior, keep humans in the loop at the right moments, and improve from realized outcomes - continuously, under a service-level objective. This paper specifies a full-stack production-platform reference architecture for systems that plan, decide, and act, whether the actor is a conversational or LLM agent, a robotic or enterprise-process automation, an industrial or IoT control loop, an operational decision-support service, or an integrated analytics platform. Every layer is a first-class, contract-bearing component: a data foundation and shared context, an orchestration and workflow runtime with saga compensation, a residency-filtered model gateway and router, an analytics layer progressing descriptive-to-prescriptive, an action-and-tool-execution layer with transactional effects, and a human-in-the-loop review surface - wrapped by two cross-cutting spines, governance and an analytics-and-observability spine, and served by a control plane under an operational-readiness discipline. The observability spine is specified in depth: capture-at-commit execution records, a composite and agent-native validity signal, trajectory-level risk and credit assignment, drift-under-action, decision-trace completeness and freshness as operational measures, escalation under a capacity model, and a closed learning loop that retrains from realized outcomes and is defended as an attack surface. Seven normative invariants (R1–R7) span the stack, and three cumulative conformance profiles (Observable, Governed, Adaptive) let heterogeneous platforms adopt the specification incrementally and interoperate. The architecture is complementary to a decision-production and conformance framework such as Governed Decision-Intelligence (GDI): that framework certifies how each decision is produced; this one specifies how the system that acts on it is built, observed, and operated.

Keywords –

reference architecture; production platform; agentic systems; enterprise automation; industrial IoT; observability; orchestration; sagas; residency-filtered routing; conformal prediction; grounding; off-policy evaluation; drift; closed-loop learning; operational readiness; conformance profiles.

Corresponding Author's: MESBAUL HAQUE SAZU

Copyright: © The Author(s), 2023. Published by **IPHO-Journal of Advance Research in Science And Engineering.** This is an **Open Access** article, distributed under the terms of the Creative Commons Attribution 4.0 License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

1 Introduction

Enterprises increasingly run systems that convert data into decisions and then into actions: an agent that resolves a support case across several tool calls; an automation that executes a back-office process end to end; an industrial platform that schedules maintenance and optimizes resources on a plant floor; a decision-support service that emits governed recommendations inside a clinical or financial workflow; an integrated analytics platform that turns operational data into revenue-risk signals. These systems succeed in a pilot under controlled conditions - contained scope, few dependencies, simple governance - and then fail to operate reliably at scale, where workflows are non-linear, dependencies are numerous, telemetry is fragmented across runtimes, and the system changes the world it is measured against.

The recurring diagnosis is that such systems are engineered as a privileged component (a model, an agent, a pipeline) with everything around it left implicit. What is missing is a full-stack reference architecture in which every layer is a first-class, contract-bearing component, tied to a shared set of testable obligations, and specified for operation in production rather than for a demonstration. This paper provides that architecture: it treats the acting system as a complete stack, wrapped by a governance spine and an analytics-and-observability spine, and served by a control plane and a deployment layer under an operational-readiness discipline of service-level objectives, error budgets, and readiness gates.

1.1 Scope: five classes of acting systems

The architecture applies to any enterprise system that plans, decides, and acts across a multi-step workflow and must be trusted while it operates. Five classes fall in scope, and the same layer contracts, decision-boundary instrumentation, and conformance profiles apply to each:

- **Conversational and LLM agents.** Multi-agent, tool-calling systems whose executions fan out into model calls, tool/API calls, and workflow logic; each execution is a trajectory carrying grounding, tool-call validity, self-consistency, and judged-quality signals.
- **Robotic and enterprise-process automation.** Platforms that turn analytics-driven decisions into executed actions with remit-based human routing and an operational loop, so a completed workflow is never mistaken for a successful outcome and analytics competency is institutionalized as a reusable capability.
- **Industrial and IoT control.** Systems whose predictions actuate physical plant - predictive risk and resource optimization - so errors have side effects and the input distribution shifts because the system acted; drift-under-action and before-commit gating are load-bearing.
- **Operational decision-support services.** Live services that emit governed recommendations under an SLO; decision-trace completeness, data freshness, readiness gates, and disaster recovery are what make them operable and auditable at runtime.
- **Integrated analytics platforms.** Multi-stage pipelines with a shared entity model that convert data into decision signals inside existing workflows and retrain from realized outcomes; capture-at-commit and the closed learning loop are their instrumentation.

Positioning relative to a decision-production framework

This architecture is deliberately complementary to, and distinct from, a decision-production and conformance framework such as Governed Decision-Intelligence (GDI). GDI governs the design-time production and certification of a single decision - a veracity gate, model-risk validation, calibration, deterministic explanation where regulated, and a decision-quality target - and answers what must be true before a decision is trusted. This paper begins where that leaves off: it takes the certified decision as given and specifies the acting system that carries it out, instrumented at the decision boundary, evaluated along the trajectory, gated before each committed action, and operated under an SLO. Many production platforms cite both.

2 Design Principles and System Invariants

The architecture rests on a small set of invariants that every conforming layer must uphold, stated as requirements (MUST/SHOULD) so conformance is testable.

Framework 1 - System invariants (normative, RFC 2119)

R1 · Capture-at-commit (MUST). Each decision freezes its basis - identity, lineage, inputs-as-references, model version, validity - at the instant of action.
R2 · Before-commit gating (MUST). A decision's validity is evaluated and governance applied before an external effect, not annotated afterward.
R3 · Compensability (MUST). Every external effect registers a compensating action or is idempotent/transactional, so trajectories can be unwound.
R4 · Provenance and temporal validity (MUST). Every context edge carries provenance, confidence, and expiry; expired or contradictory context is not served.
R5 · Residency as a hard constraint (MUST). Compliance and data-residency are feasibility filters applied before optimization, never traded against quality or cost.
R6 · Logged propensity (SHOULD). Model/action selection logs its propensity to enable off-policy evaluation.
R7 · Auditability (MUST). Every governance verdict and override is recorded with rationale on an immutable log.

These invariants recur throughout: capture-at-commit and propensity make the analytics and learning layers possible; compensability gives the escalation policy a real notion of “reversible where possible”; provenance and residency are enforced by the context, gateway, and governance layers. The conformance profiles of §14 are

defined as subsets of the layers and obligations that realize these invariants. Four reusable patterns recur across instantiations: P1 gated commit (resolve context, score, compute validity, gate before commit, then act with a registered compensation); P2 capture-and-close (write one execution record at commit, join the outcome later, promote failures to regression cases); P3 route-by-remit (send decisions beyond remit to the correct human role under a capacity model); P4 observe-and-adapt (monitor drift with a defined response, retrain on breach, off-policy-evaluate the loop before trusting it).

3 Reference Architecture Overview

The platform is organized as a central execution stack wrapped by two cross-cutting spines and served by a control plane (Figure 1). The data plane runs from ingestion and a shared-context substrate, through an orchestration and workflow runtime, a model gateway and router, an analytics layer, an action-and-tool-execution layer, and a human-in-the-loop review surface, into systems of record. The governance spine - identity, authorization, tenancy, residency, and lineage - applies in-line at every layer via a policy-enforcement point that forwards to a central policy-decision point under deny-overrides. The analytics-and-observability spine taps every layer at the decision boundary: it reads the substrate and gateway, evaluates each decision and execution, and writes back to governance and to the learning loop.

Figure 1 · Full-stack production-platform architecture — data plane, two cross-cutting spines, control plane



Figure 1 · Full-stack architecture - data plane, governance and analytics-and-observability spines, and a control plane under operational-readiness discipline.

Framework 2 - System obligations (apply at every layer)
O1 Capture-at-commit. Freeze the basis of a decision at the instant of action; the substrate beneath is mutable. **O2 Before-commit gating.** The validity signal gates the action, not annotates it after the fact.
O3 Compensability. Every committed external effect registers a compensating action; a failed or escalated trajectory unwinds in reverse. **O4 Provenance and freshness.** Every context read and write carries provenance, confidence, and temporal validity; freshness is measured.
O5 Observability. Every execution emits one joinable record; decision-trace completeness is a first-class metric. **O6 Closed loop.** Realized outcomes return as labels, regression cases, and drift triggers, screened against poisoning.

4 End-to-End Decision-and-Action Lifecycle

A single business outcome is produced by a chain of steps (Figure 2): context is resolved from the shared substrate with provenance; a model or policy is selected through the residency-filtered gateway and produces a candidate decision; the analytics layer progresses that decision from descriptive grounding to a prescriptive recommendation; the validity signal is computed and the decision is gated before commit; an action is executed with a registered compensation; the execution is captured at commit; and the realized outcome, when it arrives, closes the loop. Model a workflow as a directed acyclic graph of decision points. To each decision point d the observability spine attaches a tuple (r_d, v_d, g_d) - the execution record, the validity signal, and the governance outcome g_d ∈ {proceed, log, escalate, rewrite, halt}. Because analytics attach at the boundary and are captured at commit, compounding error can be traced from a downstream failure back to its upstream basis.

Figure 2 · End-to-end decision-and-action lifecycle — governance and analytics taps applied in-line, before commit

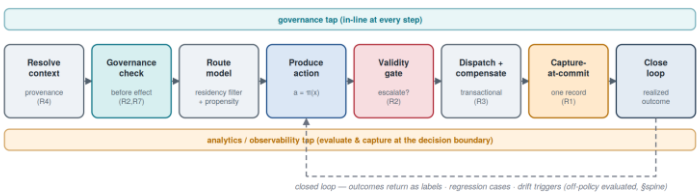


Figure 2 · One decision across the layers; governance and analytics taps are applied in-line, before commit, and the outcome closes the loop.

Taxonomy of scale failure modes. The observability spine exists to detect a family of failures that accuracy tracking does not catch in time: confidently-wrong extrapolation (a tight, plausible output outside validated support); silent covariate drift (aggregate reliability decays while per-decision behavior looks normal); ungrounded generation (a fluent free-text answer unsupported by retrieved context); compounding error across the chain (a miscalibrated upstream decision consumed as fact downstream); endogenous shift (the system's own actions move the distribution it is next measured against); and learning-loop rot and poisoning (the write-back substrate accumulates stale, contradictory, or adversarial planted context). Each layer contract is chosen to make one of these observable.

5 Layer - Data Foundation and Shared Context

The data foundation ingests structured, unstructured, and event data with origin metadata, normalizes it, and reconciles identity into a shared entity model via probabilistic record linkage; every entity key is resolved and every read is freshness-stamped. The shared-context substrate is a graph in which nodes are entities and edges are relationships and facts, and every edge carries three metadata fields: provenance (source and pipeline), confidence (a score in the unit interval), and temporal validity (the interval over which the fact holds). A retrieval-augmented read resolves the context x_d for a decision by selecting the most relevant, in-validity, sufficiently-confident edges, and the read itself is recorded so the decision can later be reconstructed against the context as it was, not as it is now. Two operational measures defined here are consumed throughout: data freshness (the age of the resolved context against a governed target per fact class - a market quote and a patient-eligibility record tolerate very different staleness) and decision-trace completeness (the fraction of a decision's basis that is captured and joinable). Contradiction, when two in-validity edges disagree, is reconciled by a defined pass or surfaced as an unresolved-contradiction rate - a substrate health metric a platform review watches directly.

6 Layer - Orchestration and Workflow Runtime

The runtime executes the decision DAG: it plans, dispatches steps or agents, passes messages, retries transient failures, and enforces ordering and concurrency. Two properties are load-bearing for the rest of the stack. Lineage: each step records its parents, so a downstream failure can be traced upstream and credit for a failure assigned across the graph. Compensability: because the system takes real actions, a partially completed workflow must be unwound, which we obtain from the saga pattern - each committed step registers a compensating action and a failed or escalated trajectory runs compensations in reverse. This gives operational meaning to the “reversible-where-possible” clause of the escalation policy: reversibility is a runtime capability the spine can query, not an assumption. For an enterprise-automation platform this layer is the core - it turns a decision into an executed action while keeping a human in the loop at defined points and judging the result by outcome rather than completion.

Algorithm 1 - Plan / Execute / Compensate
in: workflow DAG G; agents; retry policy
1: topologically order G; for each ready step d:
2: execute d; on transient failure retry per policy
3: record lineage (parents of d); register compensation C_d if effect is external
4: if d escalated or failed hard: run C_{d-1}..C_1 in reverse; halt or reroute
out: completed trajectory with lineage, or a compensated unwind

7 Layer - Model Gateway and Routing

The gateway abstracts heterogeneous models - gradient-boosted trees, linear models, and large language models - behind one interface, handling adapters, fallback, caching, and cost metering, and logging the routing propensity $p(a|x)$ needed for off-policy evaluation. Model selection is decision-dependent: a gradient-boosting default for tabular risk, an interpretable linear alternative where the precision-recall trade-off favors it, and a language model where the task is generative or conversational. Routing is a contextual bandit balancing exploitation against exploration over per-model, per-task-class estimates of quality, latency, and cost - but compliance and data-residency enter first, as a hard feasibility filter $F(x)$ applied before the quality-and-cost optimization (Figure 3). Optimizing only quality and cost would let a router send regulated data to a disallowed endpoint; making $F(x)$ a filter applied before the objective enforces invariant R5, and the chosen propensity is logged (R6) so a claim that the platform is compounding improvement is falsifiable rather than asserted.

Figure 3 · Residency-filtered contextual-bandit routing — compliance is a hard feasibility filter before optimization

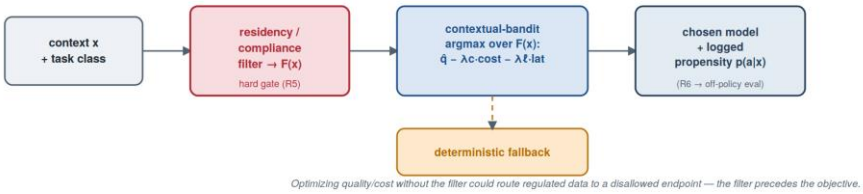


Figure 3 · Residency-filtered contextual-bandit routing; compliance is a hard feasibility filter before the objective, and propensity is metered for off-policy evaluation.

8 Layer - Action and Tool Execution

Agents and automations act through tool calls and enterprise-system connectors, and industrial platforms actuate physical plant. For the compensation guarantees to hold, each external effect must be idempotent (safe to retry, via a dedup key) or transactional (committed atomically via two-phase or an outbox) and must register a compensating action. The action layer validates each tool call before dispatch - schema conformance, precondition satisfaction, and argument ranges - which the observability spine consumes as the tool-call-validity component of §10. Where the effect is physical actuation, the before-commit gate is load-bearing: a prescriptive decision is gated before it reaches the plant, because a wrong action that auto-commits cannot be undone by re-scoring and changes the distribution the system is next measured against.

9 Layer - Governance, Identity, and Policy (spine)

Governance is a spine, not a stage: a policy-enforcement point (PEP) at each layer intercepts and forwards to a central policy-decision point (PDP) that evaluates identity, authorization, tenancy, residency, and PII predicates under deny-overrides and returns a verdict with obligations. Any deny predicate denies; permits carry obligations such as redaction or tokenization; least-privilege authorization runs under an authenticated identity with tenant isolation; and every verdict and override is recorded with rationale on an immutable log (R2, R7). This is what makes tenant isolation, residency, and audit response properties of the platform rather than per-integration afterthoughts.

10 The Analytics-and-Observability Spine

The observability spine is the layer this architecture specifies in the most depth, because it is what distinguishes a system that merely runs from one that can be trusted while it runs. Conventional observability reports that a request executed and how fast; it does not report whether the system behaved correctly, whether the output was useful, whether the run was risky, or whether it moved the decision it fed. The spine taps every layer at commit time and runs the pipeline of Figure 4.

Figure 4 · The analytics-and-observability spine — taps every layer at commit; the pipeline that makes a running system trustworthy

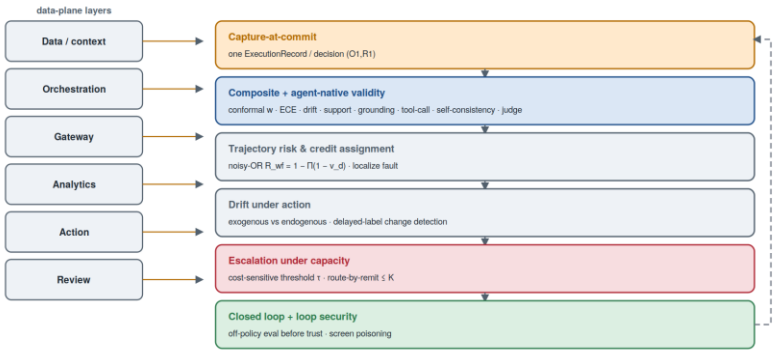


Figure 4 · The analytics-and-observability spine taps every layer at commit; the pipeline that makes a running system trustworthy.

10.1 Capture-at-commit

At the instant a decision commits, the spine writes one execution record per decision. Reconstructing a decision's basis from service logs after the fact assumes the world is immutable and fully logged; neither holds. Capture-at-commit freezes the actual basis at the moment of action, which is exactly what makes decision-trace completeness measurable and what lets a failing trace be promoted into a regression case verbatim.

```
ExecutionRecord (runtime telemetry schema) {
 decision_id, workflow_id, parent_ids // identity + lineage in the graph
 agent_id, model_version, config_hash // exactly what produced the decision
 input_refs: ContentHash[]    // inputs by reference, not copy
 candidate_actions, chosen_action  // option set and selection
 routing_propensity: float    // p(a|x) for off-policy evaluation
 policy_checks: Verdict[]    // in-line governance verdicts
 validity_signal: Validity    // classical + agent-native components
 compensation_ref: string?    // registered saga compensation
 hitl_state: enum + rationale   // escalation / override status
 trace_completeness, freshness: float // operational measures
 commit_ts: timestamp     // event-time}
```

Framework 3 · The runtime ExecutionRecord - the observability counterpart to a design-time evidence record. Full reasoning traces are gated on risk to bound storage.

10.2 A composite and agent-native validity signal

The central per-decision signal answers one question: is this the kind of input and execution on which we have evidence the system is reliable? For numeric and probabilistic outputs it fuses a conformal interval width, a calibration error, a drift term, and an out-of-support flag conjunctively, so any single strong warning drives the composite invalidity toward one and an out-of-support flag hard-overrides the rest. For free-text and tool-calling outputs, where interval width and calibration are undefined, it adds agent-native components - which is what makes the spine directly applicable to a conversational or LLM-agent platform.

Component	Applies to	What it measures
Conformal interval width	Numeric / probabilistic	Distribution-free predictive uncertainty at level $1-\alpha$
Calibration error	Probabilistic	Gap between confidence and accuracy over bins (ECE)
Distributional drift	All	Movement of inputs from validation support (PSI, KS, JS)
Support / out-of-support flag	All	Whether the input lies inside validated support (hard override)
Grounding / faithfulness	Free-text	Support of the answer by retrieved context
Tool-call validity	Tool-calling	Schema, precondition, and argument-range conformance
Self-consistency	Free-text	Agreement across independent samples
Judged quality	Free-text / agentic	Score from a calibrated model-graded evaluator

Table 1 · Validity components. Classical components fuse with agent-native components into one composite signal.

10.3 Trajectory risk, drift-under-action, escalation, and loop security

Because a single outcome is produced by a chain, the spine composes per-decision validity along the trajectory with a noisy-OR structure, $R_{wf} = 1 - \prod_d (1 - v_d)$ - the same functional form across decisions of a trajectory as across components of a decision, so a threshold set at one granularity is interpretable at the other. When a trajectory fails, credit assignment distributes responsibility across its decision points using lineage and causal-influence weights, localizing the fault to the upstream decision that most likely caused the downstream failure. Drift is treated as an operational signal with a defined response: monitored with PSI, KS, and Jensen-Shannon, a governed breach opens a retraining or revalidation action, and the shift is decomposed into exogenous (the world moved) and endogenous (the system's own actions moved the distribution) with anytime-valid change detection under delayed labels. The closed loop returns realized outcomes as labels, regression cases, and drift triggers, and its write-back is off-policy-evaluated with the logged propensities so compounding improvement is falsifiable. Escalation maps the validity signal and action stakes to a governance outcome via a cost-sensitive, capacity-aware policy: the threshold is set where the expected cost of a missed unsafe action balances the cost of a needless review, and review capacity is planned alongside it so escalation rate is a direct staffing input. Finally, the learning loop is an attack surface: the write-back substrate is screened for poisoned or adversarially planted context, and model-graded evaluators are defended against gaming.

```
Algorithm 2 - Escalation under capacity
in: validity v_d; action reversibility rev; harm cost c_harm; review cost c_rev; capacity K
1: risk <- v_d weighted by action stakes and (1 - rev)
2: tau <- threshold where c_harm*pi_unsafe*FN'(tau) balances c_rev*pi_safe*FP'(tau)
3: if risk > tau and queue < K: g_d <- escalate  // route by remit
4: elif risk > tau and queue >= K: g_d <- halt or rewrite // capacity-aware fallback
5: else: g_d <- proceed; log for the closed loop
out: governance outcome g_d
```

Storage cost. Capturing every decision is viable if the record stores references rather than payloads and gates full reasoning traces on risk. With total log storage ≈ N·(s + f·S), a compact record s of one to a few kilobytes and full traces of tens to hundreds of kilobytes gated so that f is small, tens of millions of decisions cost a few hundred gigabytes - trivially viable on tiered object storage. Storing full traces for every decision would multiply this by one to two orders of magnitude and is the design error to avoid.

11 Analytics Competency and Decision Quality as Platform Capabilities

Two capabilities platforms usually leave implicit are made first-class here because they determine whether the stack produces value. Decision quality, DQ = g(U, E), combines usefulness U - the reliability, precision, and accuracy of acted-upon outputs - with effectiveness E, the time and cost to reach the decision; acceptance and override rates are leading indicators reviewed alongside realized outcomes. Analytics competency - the reusable capability to build, interpret, and act on analytics - is institutionalized as a distinct, funded, versioned service rather than assumed to follow from tooling, and its effect on decision quality is measured through controlled or doubly-robust comparison. A platform can raise analytics use - more models, more dashboards, more automated steps - without raising decision quality, if the competency to interpret and act on the output does not rise with it; the architecture therefore instruments both, and realized decision quality as the outcome the two are meant to move. This is the difference between a platform that reports activity and one that reports value.

12 Human-in-the-Loop Review

Human judgment is routed, not bypassed. Decisions beyond automation's remit escalate to the correct role; escalation is confidence-gated and capacity-planned; and the review surface captures accept or override with rationale, feeding the write-back loop. Reviewer identity and escalation reason are under audit. The review surface is where remit and calibrated confidence jointly decide when automation hands control to a human, turning oversight into a measured control with a capacity model rather than a blanket human-in-the-loop claim.

13 Deployment and Operational Readiness

The control plane and deployment layer carry the platform under a reliability discipline. Service-level objectives and error budgets bound latency and availability; a production-readiness review gates general availability against documented exit criteria; a change-freeze policy applies when the error budget is exhausted; on-call and disaster-recovery procedures make the platform operable and recoverable. For an operational-readiness review of a live decision-support service, this layer is the review itself: each gate is assessed against documented exit criteria, the numbers are measured over a limited-availability window rather than projected, and the launch decision follows directly from the gate table.

Readiness gate	Exit criterion	Verdict
Reliability	SLOs defined and met over the LA window; error-budget policy in force	Pass / Cond. / Fail
Observability & trace	One joinable record per execution; decision-trace completeness above target	Pass / Cond. / Fail
Security & compliance	Identity, tenancy, residency filters enforced; audit trail complete	Pass / Cond. / Fail
Data & model governance	Registry, lineage, and versioning first-class; drift response defined	Pass / Cond. / Fail
Operability	On-call, runbooks, and alerting in place; escalation capacity planned	Pass / Cond. / Fail
Disaster recovery	Backups, failover, and recovery objectives tested	Pass / Cond. / Fail

Table 2 · Operational-readiness gates. A general-availability decision follows from the gate table, measured over a limited-availability window.

14 Adoption, Conformance, and Instantiation

The specification is adoptable incrementally through three cumulative, testable conformance profiles, so a platform organized around orchestration, or around analytics, or around observability can adopt it in stages and interoperate through shared execution-record and context definitions. Observable adds capture-at-commit and the composite validity signal (O1, O2, O5). Governed adds compensation, in-line governance, before-commit gating, remit-based review, and readiness gates (O3, O4, and the review and deployment layers). Adaptive adds trajectory risk, drift-under-action, closed-loop retraining with off-policy evaluation, and loop security (O6 and the full spine). Each class of §1.1 instantiates the same architecture but leads with a different subset and enters at a different profile (Figure 5).

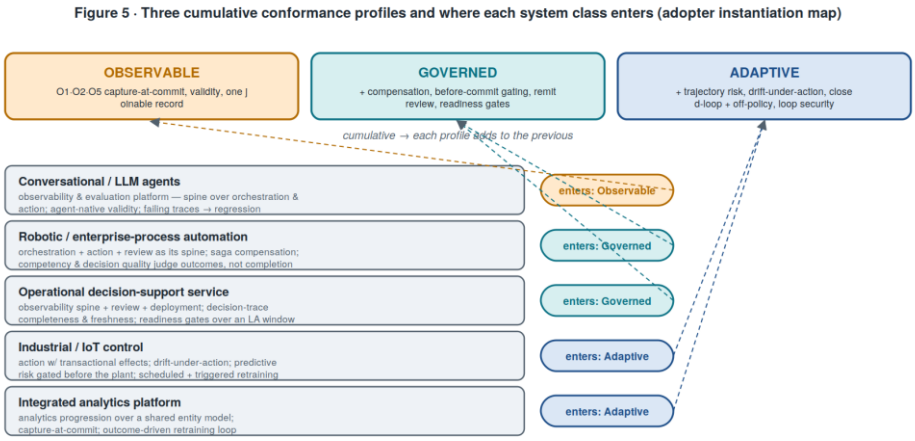


Figure 5 · Three cumulative conformance profiles and where each of the five system classes enters - the adopter instantiation map.

- **Conversational / LLM agents.** An observability-and-evaluation platform instantiates the spine (§10) over orchestration and action: one record per execution, a versioned evaluation service decoupled from agent runtimes, agent-native validity and trajectory risk, and a loop that promotes failing traces into regression cases. Enters at Observable.
- **Robotic / enterprise-process automation.** An automation platform instantiates orchestration, action, and review as its spine - saga compensation, transactional effects, remit-based routing - with competency and decision quality judging realized outcomes rather than completion. Enters at Governed.
- **Operational decision-support service.** A live service instantiates the observability spine, review surface, and deployment layer: decision-trace completeness and freshness as operational measures, escalation under capacity, and readiness gates over a limited-availability window. Enters at Governed.
- **Industrial / IoT control.** A platform that actuates plant instantiates the action layer with transactional effects and leans on drift-under-action: predictive risk and resource optimization gated before reaching the plant, with scheduled and triggered retraining as conditions shift. Reaches Adaptive.
- **Integrated analytics platform.** A layered pipeline over a shared entity model instantiates the analytics progression and the closed loop: descriptive and text analytics into predictive risk signals into prescriptive decision support, with capture-at-commit and outcome-driven retraining. Reaches Adaptive.

15 Evaluation, Discussion, and Threats

A platform is evaluated on operational and decision metrics together: latency and availability against the SLO and error budget; step and tool success; decision-trace completeness and freshness; drift caught and actioned; escalation rate against planned capacity; and decision quality - usefulness, effectiveness, acceptance, and override - measured against realized outcomes rather than assumed.

System class	Leading operational and decision metrics
LLM / conversational agents	Per-execution validity; grounding & tool-call validity rates; trajectory risk; regression-case yield from failing traces
Process automation	Straight-through rate; escalation rate vs capacity; override rate; realized outcome quality vs completion
Industrial / IoT control	Drift caught and actioned; endogenous-shift share; before-commit gate rate; retraining latency
Decision-support services	Availability vs SLO; error budget consumed; decision-trace completeness; freshness; escalation rate
Integrated analytics platforms	Signal precision at fixed alert budget; loop retraining cadence; off-policy improvement estimate

Table 3 · Leading metrics by system class. All are measured against realized outcomes, not workflow completion.

Three implications follow from making the observability spine first-class. Validity becomes architectural rather than optional: a system that acts on a confidently-wrong output is unsafe precisely at the boundary where the action commits, so the spine gates before commit. Improvement becomes a claim the platform can test rather than assert: logged propensities and joined outcomes turn compounding improvement into a measured quantity, and a failing trace becomes a regression case that prevents the same failure twice. Human judgment is placed by

remit and calibrated confidence with a capacity model. Threats remain: the specification is a reference architecture, not a controlled causal study; its composed instruments - sagas, conformal prediction, contextual-bandit routing, off-policy evaluation, drift statistics - are not individually novel, the contribution being their integration into one conformance-testable full-stack platform with a first-class observability spine; and outcome ownership may be external, limiting closed-loop learning where downstream systems do not expose timely labels.

16 Conclusion

This paper specified a full-stack production-platform reference architecture for systems that plan, decide, and act - agents, automations, industrial controls, decision-support services, and integrated analytics platforms. Every layer is a first-class, contract-bearing component; two spines, governance and an analytics-and-observability spine, apply throughout; and a control plane operates the whole under a readiness discipline. The observability spine - capture-at-commit, a composite and agent-native validity signal, trajectory risk, drift-under-action, a closed learning loop, and escalation under capacity - is what lets a platform be trusted while it runs. The architecture is complementary to a decision-production and conformance framework: that framework certifies how each decision is produced; this one specifies how the system that acts on it is built, observed, and operated. Three cumulative conformance profiles let heterogeneous platforms adopt it incrementally and interoperate.

References

1. Sigelman, B. et al. Dapper, a Large-Scale Distributed Systems Tracing Infrastructure; OpenTelemetry specification.
2. Garcia-Molina, H. and Salem, K. Sagas (compensating transactions for long-running work).
3. Lewis, P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.
4. Li, L. et al. A Contextual-Bandit Approach to Personalized Recommendation; Dudík, Langford, and Li, doubly robust policy evaluation.
5. Vovk, V., Gammerman, A., and Shafer, G. Algorithmic Learning in a Random World; split-conformal prediction.
6. Population Stability Index; Kolmogorov–Smirnov statistic; Jensen–Shannon divergence; Ramdas et al., anytime-valid inference.
7. Manakul, P. et al. SelfCheckGPT; Es, S. et al. RAGAS; Zheng, L. et al. Judging LLM-as-a-Judge (model-graded evaluation).
8. Greshake, K. et al. Indirect Prompt Injection; Biggio, B. et al. Poisoning Attacks (learning-loop security).
9. Beyer, B. et al. Site Reliability Engineering: service-level objectives, error budgets, production-readiness review.
10. Sculley, D. et al. Hidden Technical Debt in Machine Learning Systems.
11. CRISP-DM: the descriptive-to-prescriptive analytics lifecycle; analytics-capability and decision-quality model.
12. Governed Decision-Intelligence (GDI): a decision-production and conformance framework (companion work).