Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.082270

REVIEW

Auditable LLM Autonomy for Operational Decision-Making: Big Data Evidence and Decision Traces

Leonidas Theodorakopoulos and Alexandra Theodoropoulou*

Management Science and Technology, University of Patras, Patras, Greece

*Corresponding Author: Alexandra Theodoropoulou. Email: theodoropouloua@upatras.gr

Received: 12 March 2026; Accepted: 13 May 2026; Published: 15 June 2026

ABSTRACT: Auditable autonomy is becoming a practical requirement for deploying large language model (LLM) agents in operational workflows where recommendations can trigger consequential actions. Many autonomy claims remain hard to evaluate because studies emphasize task completion or fluent explanations while underreporting tool privileges, verification conditions, rollback feasibility, and trace completeness. This review develops a decision-making–centered framework that treats autonomy as an auditable engineering property. It introduces a three-plane big data foundation: an evidence plane with provenance and freshness constraints; a decision-trace plane that records retrieval identifiers, tool invocations, intermediate checks, and policy evaluations; and an outcomes plane that captures post-decision effects such as KPI shifts, rollback incidence, recurrence, and operator overrides. On this basis, we present an assurance stack taxonomy covering grounding controls, verification and precondition checks, action gating for risk bounding, and accountability mechanisms. We then propose an evaluation ladder that moves beyond text metrics toward evidence fidelity, action validity, and longitudinal stability, with reporting requirements that improve cross-domain comparability. Finally, we synthesize accidental failures and adversarial threats, including staleness, drift, prompt injection, partial execution, and trace tampering, and map trace-based detection signals and mitigations to the assurance stack.

KEYWORDS: LLM agents; big data analytics; artificial intelligence; autonomous agents; decision support systems; auditability; automation; data governance

1 Introduction

Large language models are increasingly embedded in operational settings where the output is not merely an explanation or a report, but an input to consequential action. In many real deployments, agentic systems are already being asked to diagnose incidents, assemble evidence across heterogeneous data stores, propose remediation steps, open or update workflow artifacts (for example, tickets and change requests), and in some cases trigger tool-mediated actions. That shift matters because it changes the evaluation target. Fluency and apparent plausibility become secondary once a system's recommendations can alter production states, affect safety margins, expose sensitive resources, or create latent reliability degradation that is only detected after the fact [1].

A central difficulty is that many claims of LLM autonomy in operational settings remain hard to evaluate in a decision-relevant way. Studies often emphasize task completion, plan plausibility, or response quality, while giving much less attention to the conditions that determine whether an operational recommendation is actually defensible: the authority and freshness of the evidence used, the verification steps performed, the

Copyright © 2026 The Authors. Published by Tech Science Press. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

constraints on tool-mediated action, and the completeness of the resulting decision trace [2,3]. As a result, autonomy can appear strong at the interface level while remaining weakly bounded, weakly auditable, or difficult to compare across studies.

The central claim of this paper is that autonomy in operational workflows is only defensible when it is auditable. Auditability, in this context, is not a narrative property, and it is not satisfied by post-hoc explanations detached from verifiable evidence. It is an operational property that must be designed into the system: the system must produce a reconstructable decision trace, show that its action proposals were conditioned on bounded and time-relevant evidence, and operate under explicit constraints that prevent high-impact actions from being taken when verification is incomplete. Without these properties, agentic operations risk reproducing familiar failure patterns from automation (overreach, brittle generalization, silent error propagation), while adding new failure modes tied to tool use, retrieval errors, and non-deterministic multi-step reasoning.

1.1 Agentic Operations as a Decision Problem, Not a Language Problem

Operational environments differ from standard question-answer settings in ways that undermine common LLM evaluation habits. The system rarely observes the full state of the environment. Evidence is fragmented across logs, metrics, traces, configuration histories, tickets, runbooks, and asset inventories, each with different latency, authority, and error characteristics. The same surface symptom can map to different root causes, and the cost of an incorrect action can dominate the benefit of multiple correct suggestions. These conditions push the problem into decision-making under uncertainty, where the system's value depends on how it manages incomplete evidence, how it bounds the consequences of its proposals, and how it behaves when verification fails [4].

A second distinction is that agentic systems introduce a wider action surface. Once a model is connected to tools, it can interact with production systems, data stores, and workflow platforms. This expands both opportunity and risk. The most relevant question becomes whether the system can be trusted to act within constraints that reflect operational realities: least privilege, staged escalation, change control, separation of duties, and safe rollback. In other words, the important unit of analysis is the decision and its operational footprint, not the generated text [5].

1.2 Why "Decision Assurance" Is the Missing Center of Gravity

The current literature contains valuable work on LLMs for operations, tool-using agents, and LLM security. Yet much of it remains capability-forward: can the system complete a task, retrieve relevant context, or produce a plausible plan. This focus is understandable in an early phase of technology diffusion, but it leaves an analytical gap. Capability does not imply controllability, and controllability is precisely what operations require [6].

This paper uses decision assurance to name the set of mechanisms that make operational autonomy defensible. Decision assurance is narrower than general safety discussions and more technical than governance statements that remain at the policy level. It concerns the controls that sit between evidence and action: how evidence is qualified, how preconditions are verified, how actions are bounded by enforceable rules, and how the system's internal decision path is captured in a trace that can be inspected later. A critical implication follows: assurance cannot be treated as an optional layer added after an agent works. In action-taking contexts, assurance is part of what it means for the system to work at all [7].

A related challenge is the tendency to conflate explanation with accountability. Explanations can be persuasive while still being wrong, incomplete, or unsupported by authoritative data. Auditability requires

a different standard: the system must be able to show what it used, when it used it, which constraints were checked, what was executed (or not executed), and how outcomes were monitored. That standard is not a stylistic preference; it is the basis for error correction, post-incident learning, and organizational accountability [8].

1.3 Big Data Traces as the Foundation of Auditable Autonomy

Auditability depends on data, but not in the generic sense of volume alone. In agentic operations, the relevant issue is whether the system leaves a usable operational record that links what it saw, what it did, and what followed from the decision. In this review, that record is framed through three connected planes: an evidence plane that captures the operational basis of the decision, a decision-trace plane that records how the recommendation or action was assembled and constrained, and an outcomes plane that records what happened after the recommendation was adopted or the action was executed [9].

This framing shifts big data from background context to an assurance substrate. If evidence cannot be linked to claims, if decision traces are incomplete, or if outcomes are not recorded in a way that supports later review, then auditability remains largely rhetorical. The same framing also clarifies why assurance and evaluation cannot be separated: assurance mechanisms determine what is recorded and what can later be reconstructed or tested [9].

At the same time, trace-rich systems introduce new governance burdens. Expanded trace collection increases the sensitive-data footprint, creates retention and access-control demands, and can itself become an attack surface if evidence stores or trace records are manipulated. For that reason, auditability must be treated as both a reliability requirement and a security requirement, not merely as a documentation convenience [10,11].

1.4 Contributions and Positioning of This Review

This article is a comprehensive review and critical synthesis focused on auditable autonomy for LLM-based agentic operations. Rather than cataloging applications, it organizes the literature around the mechanisms and measurement practices that determine whether operational autonomy is safe to deploy and defensible to audit.

The main contributions are:

1. an assurance-oriented taxonomy that separates evidence grounding, verification, action gating, and auditability into distinct control layers, each with different failure modes and implementation burdens.
2. a big data trace framing that treats operational evidence, agent decision traces, and outcomes as a linked enabling basis required for accountability and for decision-level evaluation.
3. an evaluation perspective that prioritizes decision quality and operational risk, including reversibility, rollback costs, unsafe-action rates, and trace completeness, over text-level metrics.
4. a set of synthesis artifacts (figures and tables) designed to make claims comparable across studies, and to expose where published evaluations omit critical operational details.

The goal is not to argue that full autonomy is generally achievable or desirable. Instead, the review takes a constrained position: operational autonomy should be considered credible only to the extent that it is risk-bounded, evidence-grounded, and auditable, with evaluation methods that reflect those requirements.

1.5 Organization of the Paper

Section 2 defines the conceptual boundaries of agentic operations, decision assurance, and big data traces, and separates this paper's focus from adjacent surveys that center on capabilities or general LLM

safety. Section 3 describes the review methodology and how the main corpus was assembled for a comprehensive, non-systematic synthesis. Section 4 introduces a taxonomy of operational decision types and action surfaces, with an emphasis on risk, reversibility, and the practical meaning of "correctness." Section 5 develops the big data trace foundations needed for auditable autonomy, distinguishing operational evidence, agent decision traces, and outcome records. Section 6 surveys common agentic architectures in operational workflows and identifies where assurance controls attach in practice. Section 7 presents the core taxonomy of decision assurance mechanisms, with an explicit treatment of trade-offs and residual risks. Section 8 focuses on evaluation designs and metrics that measure decision quality and operational risk rather than text quality. Section 9 examines failure modes and adversarial considerations, linking them back to assurance controls and trace signals. Section 10 synthesizes cross-domain implications and limits to generalization. Section 11 outlines a focused research agenda tied to traceability, evaluation, and governance constraints. Section 12 closes with the implications of evaluating autonomy through auditability, traceable evidence and enforceable constraints rather than through narrative plausibility alone.

2 Background and Conceptual Boundaries

This section fixes the conceptual boundaries that the rest of the paper assumes. The aim is to avoid re-litigating definitions later and to keep the technical argument focused on assurance controls, traceability, and evaluation.

In this review, trace completeness and trace sufficiency are treated as related but different properties. *Trace completeness* refers to whether the recorded decision trace contains the minimum linked artifacts needed for later reconstruction and audit, including the evidence references used, the checks performed, the tool interactions attempted, the final recommendation or action, and the relevant constraint context.

Trace sufficiency refers to whether those recorded artifacts show that the decision was supportable at the time it was made, given the authority, freshness, coverage, and conflict status of the available evidence. A trace can therefore be complete but still reveal an insufficient basis for action, for example when key signals were missing, stale, or contradictory. The assurance stack is used here in a dual sense: descriptively, as a synthesis device for organizing the literature into comparable control layers, and normatively, in the weaker sense that credible autonomy claims should specify how the relevant layers are implemented or why they are not required for the decision surface under study.

2.1 Operational Decision-Making and Action Surfaces

Operational decision-making is treated here as an activity that couples interpretation of imperfect signals with commitments that change system state or allocate organizational attention. This definition is intentionally narrower than "reasoning about an incident" because the review is concerned with decisions that create downstream obligations: altering configurations, initiating containment, triggering escalations, opening change requests, or assigning on-call actions [12]. In agentic settings, the decisive shift is that recommendations are often produced in a form that is already executable within a tool ecosystem, even if final execution remains gated by human approval [13].

A useful boundary is between decision intent and action surface. Decision intent captures what the decision is for, such as diagnosis, prioritization, or prescription. Action surface captures where the decision lands, such as change management, incident workflows, security controls, data pipeline operations, or resource orchestration. The same intent can map to very different risks depending on the surface [14]. A diagnosis error may be recoverable when it only delays triage, but it becomes high-impact when it triggers automated containment or rollback. Similarly, "propose remediation" is not equivalent to "execute

remediation," even when the text appears identical; the operational meaning changes once a tool call is available [15].

Two recurring constraints shape decision quality in operational environments. The first is partial observability: evidence is fragmented and frequently ambiguous, and absence of evidence is often the normal case rather than an anomaly [16]. The second is cost asymmetry: false positives and false negatives rarely carry symmetric consequences, and error costs can compound through dependencies. These constraints matter because they invalidate simplistic evaluation practices that focus on task completion or textual plausibility [17]. A defensible autonomy claim must therefore be grounded in an explicit characterization of the decision surface and the consequences of error, including reversibility and blast radius.

To avoid later repetition, Sections 4 and 5 will use this boundary consistently: intent and surface will be treated separately, and risk will be tied to surface-specific consequences rather than treated as an vague property of the agent.

2.2 Big Data Traces as an Evidentiary Foundation for Auditable Autonomy

The phrase "big data traces" is used here to describe the linked operational record that makes agentic decisions inspectable at scale. It includes the evidence consulted at decision time, the trace of how the recommendation or action was produced, and the observed outcomes that followed [18,19]. These three planes provide the minimum evidentiary structure needed for reconstruction, review, and evaluation.

This linked record matters because auditability cannot be established through explanation alone. A system may provide a plausible narrative while still failing to show which sources were consulted, which checks were performed, what was executed, or how later outcomes were monitored. For that reason, the paper treats trace design as part of assurance engineering rather than as a secondary implementation detail [20,21].

The detailed role of each plane is developed in Section 5. Here, the point is narrower: auditable autonomy requires more than successful outputs. It requires an inspectable record that connects evidence, constrained decision process, and post-decision effects in a form that can support later scrutiny.

2.3 Distinguishing This Review from Adjacent Literatures

Several research strands intersect with this topic, but they do not fully address the same problem.

First, generic LLM safety and responsible AI discussions often emphasize content harms, bias, privacy leakage, or misinformation [22–24]. These concerns remain relevant, but they do not directly resolve the question of action correctness under operational uncertainty, nor do they specify enforceable constraints on tool-mediated actions. Operational autonomy has failure modes that are not well captured by content-focused safety taxonomies, particularly when errors manifest as silent service degradation, misconfigured controls, or inappropriate escalation [22].

Second, the AIOps and LLM-for-operations literature frequently catalogs use cases and model families, sometimes with promising demonstrations. What is often missing is a comparable accounting of decision assurance mechanisms: what preconditions were verified, what actions were permitted, what was blocked, what trace data was retained, and how unsafe actions were identified and measured. When these details are absent, reported successes are difficult to interpret and hard to transfer [25,26].

Third, security work on tool-using agents and prompt injection has advanced quickly and offers a useful threat lens. However, security analyses alone do not yield a complete assurance approach because they typically focus on adversarial inputs and exploit paths, while operational assurance must also address non-adversarial failure: staleness, ambiguity, conflicting evidence, tool partial failure, and drift. A defensible

approach requires a unified view in which security constraints, verification checks, and auditing practices jointly bound operational risk [27].

This paper therefore treats auditable autonomy as a constrained operational objective that integrates these strands but is not reducible to any one of them. The remainder of the review builds from this boundary: operational decisions are analyzed by intent and action surface; auditability is treated as a property of traceable evidence and constrained action; evaluation is framed as decision quality under uncertainty rather than narrative quality.

3 Methods

This paper is written as a comprehensive review and critical synthesis. The objective is to integrate several partially overlapping literatures, namely agentic systems, operational decision assurance, auditability/traceability, and evaluation beyond text, into a coherent set of concepts, mechanisms, and measurement practices. It is not treated as a systematic literature review: the goal is not exhaustive coverage with formal PRISMA-style accounting, but an analytically grounded corpus that supports a defensible synthesis and enables critical comparison across approaches.

3.1 Review Type and Rationale

A systematic approach is well suited when the field has stable definitions, comparable study designs, and sufficiently standardized outcomes to support near-exhaustive retrieval and structured aggregation. The present topic does not meet those conditions. Terminology varies across communities (operations, security, software engineering, agent research), evaluation regimes are heterogeneous, and many papers do not report the operational details necessary to compare autonomy claims (tool privileges, verification steps, gating rules, trace retention). Under these constraints, a comprehensive review is the more appropriate choice: it allows the analysis to focus on what is transferable, what is underspecified, and what appears overstated, while also acknowledging that the resulting corpus is analytically constructed rather than exhaustively retrieved in the PRISMA sense. For that reason, the review emphasizes transparency of search logic, screening criteria, and synthesis rationale, while recognizing the possibility of selection bias inherent in a non-systematic design.

3.2 Search Strategy and Sources

Search and screening were conducted across publisher platforms that commonly host peer-reviewed work indexed in Scopus and Web of Science: ACM Digital Library, Elsevier/ScienceDirect, SpringerLink, Wiley Online Library, and ACL Anthology (primarily for retrieval-grounding and evaluation resources). Standards and governance materials were used selectively as normative anchors for auditability and risk management (for example, NIST publications), but the empirical and technical claims are grounded in the peer-reviewed corpus rather than in policy documents or vendor commentary.

Search strings were iterated to capture lexical variation across communities. Query families combined terms for LLM agents and tool use (agent, tool-using, function calling, RAG, autonomous, multi-step) with terms for operational settings and assurance mechanisms (operations, incident response, change management, SRE, SecOps, verification, gating, auditability, observability, tracing, evaluation). Screening emphasized whether a paper engages a genuine decision surface and whether it reports enough detail to support critical synthesis, not only whether it mentions “agents”.

3.3 Inclusion and Exclusion Criteria

The criteria were designed to ensure that the main corpus supports the paper's central constructs: decision assurance, action gating, and big data traces as the evidentiary basis for auditable autonomy. Exclusions are explicit to avoid ambiguity, especially regarding preprint repositories and non-scholarly sources. Table 1 summarizes the inclusion and exclusion criteria of this review paper.

Table 1: Summary of inclusion and exclusion criteria.

Criterion Type	Rule	Practical Interpretation for Screening
Inclusion	Peer-reviewed journal article or established conference proceeding available through major scholarly publisher platforms or recognized disciplinary venues	The item appears on major publisher platforms (ACM, Elsevier, Springer, Wiley) or ACL Anthology, and is not solely a preprint listing.
Inclusion	Clear connection to agentic systems or tool-augmented LLM workflows	The system performs multi-step reasoning with retrieval and/or tool interaction, not only standalone text generation.
Inclusion	Presence of an operational decision surface	The work connects model outputs to actions such as remediation planning, escalation, configuration change, containment, workflow automation, or equivalent operational commitments.
Inclusion	Substantive discussion of assurance mechanisms, constraints, or audit/trace practices	The paper specifies controls such as precondition checks, policy constraints, privilege boundaries, staged automation, trace instrumentation, or reproducibility measures.
Inclusion	Evaluation includes or enables decision-level interpretation	The evaluation reports outcomes that speak to operational decision quality, risk, reliability, security, or trace completeness, even if imperfectly.
Inclusion	Transferable methods supporting assurance even when the domain is not "operations"	Papers on agent evaluation, traceability, retrieval faithfulness, or tool-use security are included when they contribute mechanisms or measurement concepts that can be mapped to operational assurance.
Exclusion	Non-scholarly sources as primary evidence	Blogs, marketing whitepapers, and vendor posts are not used to justify core claims. If mentioned at all, they are treated as contextual signals and not as evidentiary support.

(Continued)

Table 1 (continued)

Criterion Type	Rule	Practical Interpretation for Screening
Exclusion	"LLM application" papers with no operational action surface	Pure chat assistants, summarization tools, or QA systems that do not connect to operational decisions are excluded unless they contribute directly to assurance or evaluation mechanisms.
Exclusion	Papers lacking enough methodological detail to support synthesis	Work that does not describe tool access, data sources, constraints, or evaluation conditions is deprioritized because it prevents critical comparison and encourages over-interpretation.
Exclusion	Narrow domain case reports that cannot be generalized and do not expose mechanisms	Single-organization anecdotes without transferable controls, trace practices, or evaluation logic are excluded from the main corpus, though they may be noted as illustrative limitations in discussion.

3.4 Corpus Construction and Synthesis Approach

Selection proceeded in two passes. The first pass identified a core corpus of works that directly address agentic decision-making with operational actions, assurance controls, traceability/observability, or rigorous evaluation of tool-using agents. The second pass added a supporting corpus for methods that are essential to the argument but not always labeled as "ops," such as agent benchmarking, retrieval faithfulness evaluation, and security analyses of tool-using LLM agents. Supporting works are used to sharpen critique and to identify where operational papers make untested assumptions, rather than to inflate breadth.

Synthesis used a structured coding scheme aligned to the paper's analytic backbone:

1. Decision characteristics: intent, action surface, reversibility, consequence scale.
2. Assurance mechanisms: grounding, verification, gating, auditability controls, including reported trade-offs.
3. Trace and data requirements: what evidence is cited, what decision traces are recorded, how outcomes are linked.
4. Evaluation design: offline replay, controlled rollout, stress testing, metrics and confounders.

This approach is intentionally judgmental: it treats missing operational details (tool privileges, gating policy, verification steps, trace retention) as substantive limitations rather than neutral omissions, because such omissions directly affect whether autonomy claims can be audited or compared.

This design inevitably involves judgment in corpus construction, and it does not claim exhaustive retrieval in the manner of a formal systematic review. The purpose, instead, is to assemble a transparent and analytically useful body of literature for comparing assurance mechanisms, trace practices, and evaluation designs across partially overlapping fields. To reduce arbitrariness, the review relied on explicit inclusion and exclusion criteria, mainly major peer-reviewed publisher platforms, and a structured coding scheme aligned to the paper's central constructs. Even so, the possibility of selection bias remains and should be understood as a limitation of the review design.

4 Operational Decision Spaces and Risk Classes

This section links agentic operation claims to the concrete decisions they are meant to support. It is difficult to evaluate autonomy as a general term because the same model behavior can be benign in one decision space and unacceptable in another. Action gating and audit requirements differ largely because decision types differ in consequence, reversibility, and the practical feasibility of verification [28]. The analysis therefore starts from the decision itself, then characterizes risk in a way that is usable for later assurance taxonomies and evaluation metrics.

4.1 Decision Intent Taxonomy

Decision intent describes what the decision is trying to accomplish, independent of the tool chain used to carry it out. This separation matters because the same underlying intent can be implemented through very different workflows across organizations, while the assurance burden is usually determined by the intent and its consequence profile rather than by tooling preferences [29,30].

Diagnosis decisions aim to characterize what is happening and why. In operational settings, the output is rarely a single root-cause statement. It is typically a ranked set of hypotheses tied to evidence, along with a statement of uncertainty and, ideally, suggestions for what evidence would reduce ambiguity [31]. The critical issue for agentic systems is not whether they can produce a plausible explanation, but whether they can avoid over-committing when evidence is incomplete, contradictory, or stale. A diagnosis that looks coherent but is unsupported by authoritative signals is a precursor to unsafe downstream prescriptions, and it becomes difficult to correct if it is not traceable to the evidence used at decision time.

Prioritization decisions allocate attention and resources. They include triage severity assignments, escalation decisions, and the ordering of investigative steps. These decisions often appear low risk because they do not directly change system state, but they can be operationally consequential when they delay mitigation, trigger unnecessary paging, or normalize repeated degradation [32]. In many environments, prioritization is where human judgment is most entrenched and where automation bias can be subtle: a system that consistently ranks certain classes of incidents as “low priority” can shape organizational learning and incident response posture over time.

Prescription decisions propose actions intended to improve system state, reduce exposure, or restore service. They might recommend configuration changes, containment steps, mitigation procedures, or alternative workflows. Prescription is the point where a language-centric evaluation becomes especially misleading: the same remediation suggestion can be operationally correct, harmful, or irrelevant depending on context that is not visible in the text. For this reason, prescriptions are only defensible when they are tied to precondition checks and bounded by enforceable constraints on what the system is permitted to propose or initiate [33].

Execution decisions commit actions through tools, automation hooks, or workflow systems. The defining property is not that the agent calls an API, but that a decision is operationalized in a way that can have immediate effects and that can partially succeed or fail in complex ways. Execution therefore introduces new failure classes: partial execution, tool-side inconsistencies, silent permission denials, and sequencing errors that are not evident in the natural language trace [34]. If auditability is weak at this stage, post-incident reconstruction becomes speculative, and unsafe actions are hard to attribute to specific evidence or checks.

Governance decisions manage how decisions are allowed to occur. They include approval routing, change scheduling, policy enforcement, access control, and rollback authorization. Governance is often treated as external to the AI system, but in agentic operations it becomes part of the decision pipeline because gating rules and approval logic determine which actions can be taken under uncertainty. A governance

decision is therefore an assurance mechanism in itself, and it should be evaluated as such, not only as a compliance overlay [35].

A practical implication of this taxonomy is that a single agent may span multiple intents in one workflow, often without explicit separation. That creates hidden coupling: an overconfident diagnosis can lead to premature prescription; a flawed prioritization can suppress evidence collection; an execution step can make later diagnosis harder by changing signals [36]. For later sections, it is therefore useful to treat intent boundaries as design boundaries, even if the implementation is unified.

4.2 Risk and Reversibility Classification

Risk in operational decision-making cannot be reduced to an error rate because the same error class can have radically different consequences depending on reversibility, blast radius, and time sensitivity [37]. Correctness is therefore contextual rather than absolute.

Reversibility captures whether a decision or action can be undone with predictable effort and bounded side effects. Some actions are reversible in principle but not in practice [38]. A configuration change may have a documented rollback path, yet rollback can still trigger cascading failures, violate compliance constraints, or require downtime windows [39]. Conversely, some decisions that do not directly alter system state can still be difficult to reverse because they shape human behavior and organizational response. Repeated under-escalation, for instance, can normalize degraded service and weaken incident learning [40].

Blast radius describes the scope of impact if a decision is wrong. That scope is not only technical, such as one service vs. a shared platform, but also organizational and temporal. A flawed action may affect one team, several dependent teams, or create latent risk that becomes visible only later. Agentic operations can widen the effective blast radius because they enable faster, more frequent actions and often operate through shared tooling with broad privileges [41,42]. This is why action gating cannot be uniform across decision intents: a mechanism that is adequate for a low-radius action may be unacceptable for shared infrastructure.

Time sensitivity describes how quickly a wrong decision becomes costly and how much time is available for verification. Urgency does not remove the need for verification; it changes what kind of verification is feasible [43]. Under tight time constraints, systems may need cheaper checks, staged actions, and stronger rollback readiness. A fast response that cannot show what was checked and why is not only risky; it is also hard to audit and difficult to improve.

These dimensions show why correctness must be evaluated in context. A technically correct prescription that arrives too late may be operationally ineffective [44]. A correct diagnosis without evidentiary linkage is not auditable and may be indistinguishable from a plausible fabrication during postmortem review [45]. A correct execution that violates governance constraints may still be unacceptable [46]. For that reason, evaluation must include consequence-weighted metrics and trace completeness, as Section 8 later argues.

The same classification also explains why assurance cannot be reduced to a single control. Low reversibility and high blast radius call for stronger gating, tighter privilege boundaries, and stricter trace requirements [47]. High time sensitivity, by contrast, pushes systems toward staged automation, lightweight but explicit verification, and rapid rollback preparedness rather than unrestricted execution [48]. These are structural constraints, not stylistic preferences, and they provide a rationale for the assurance mechanisms discussed in later sections.

Table 2 below is designed to be used later as a reference point for assurance mechanisms and evaluation choices. It intentionally avoids overly granular subcategories; the goal is to keep the mapping interpretable and adaptable.

Table 2: Decision intent, action surface, reversibility, and acceptable automation mode.

Decision Intent	Illustrative Action Surfaces	Typical Reversibility Profile	Acceptable Automation Mode (Baseline)
Diagnosis	Incident investigation, health assessment, anomaly characterization	High if treated as hypothesis generation; lower if diagnosis triggers downstream actions automatically	Recommendation with traceable evidence and explicit uncertainty; automation only if downstream actions are gated separately
Prioritization	Triage severity, escalation routing, on-call paging, work queue ordering	Mixed; operationally difficult to undo if it changes attention allocation under pressure	Recommendation with human confirmation for high-impact escalations; automation acceptable for routing when reversibility and overrides are explicit
Prescription	Remediation planning, mitigation selection, containment proposals, change suggestions	Variable; can be low when changes affect shared systems or compliance posture	Recommendation with precondition checks and policy constraints; supervised execution only for low-radius, reversible actions
Execution	Tool-mediated changes, rollback initiation, resource scaling, containment actions	Often low in practice due to partial execution and side effects	Limited autonomy only for tightly bounded, reversible actions with monitoring; otherwise, supervised execution with approval workflows
Governance	Approval logic, change windows, access control, separation of duties, rollback authorization	High if policy is well designed, but errors can have systemic implications	Automation acceptable for policy enforcement and trace capture; policy changes should require human control and review

5 Big Data Foundation for Auditable Autonomy

Building on the earlier conceptual framing, this section develops the three-plane record that underpins auditable autonomy in operational settings. The analytic purpose is practical rather than descriptive: to specify what must be recorded if a system's recommendations or actions are to be justified, reconstructed, and evaluated after the fact [49–52].

The three planes serve different but connected functions. The evidence plane captures the operational basis on which a decision rests. The decision-trace plane records how that basis was translated into a recommendation or action under specific checks and constraints. The outcomes plane records what followed once the recommendation was adopted or the action was executed. The value of the framework lies not only in the planes themselves, but in the recorded links between them, because those links make later auditing, debugging, and qualified outcome evaluation possible.

5.1 Evidence Plane: What the Agent Is Allowed to Claim as Support

The evidence plane contains the operational data sources that an agent may cite when forming a diagnosis, prioritization, prescription, or governance recommendation. The core assurance issue is not evidence volume, but evidentiary strength. Operational sources differ in authority, freshness, and failure mode, so an agent is only evidence-grounded if it can treat these differences explicitly rather than folding all retrieved material into a single undifferentiated context [53].

Time-series metrics are often the fastest and most standardized signals, but they are rarely self-explanatory. A latency spike, error-rate increase, or resource saturation may be consistent with several causal stories [54]. Metrics are therefore useful for detection and monitoring, yet weak on their own for attribution. Over-reliance on metrics can encourage spurious causal narratives, especially when the system is rewarded for producing one coherent explanation instead of representing uncertainty.

Logs and distributed traces offer richer context, but they are not neutral records. Logging coverage is uneven, traces are sampled, and instrumentation changes over time. In addition, the meaning of a log line depends on software version, deployment state, and local developer practice. Agents that treat textual logs as authoritative without checking versioning, context, or sampling assumptions may produce confident but brittle decisions [55–57].

Configuration histories, change diffs, and infrastructure state are often closer to authoritative ground truth for what changed. Even so, their reliability depends on the governance processes that produced them. Shadow changes, out-of-band interventions, and inconsistent configuration stores weaken their authority [58]. The practical point is that many operational environments are not fully controlled, and an agent that assumes otherwise will misread evidence and understate risk.

Tickets, runbooks, and postmortems contain useful human judgment and organizational memory, but they should not be treated as evidence in the same sense as current system state. They encode conventions, local terminology, and sometimes outdated procedures. Their value is interpretive and procedural. They can guide investigation, but they should still be checked against the current environment before being used as support for action [59].

Across all of these sources, two constraints remain central. The first is partial observability: missing signals are normal, not exceptional, and systems should resist converting sparse evidence into confident narratives. The second is non-stationarity: the meaning and reliability of evidence drift as systems, architectures, and workflows change. A runbook written for an earlier architecture may be less useful than no retrieval at all [60]. For that reason, a defensible evidence plane requires explicit provenance and freshness controls, along with a clear distinction between authoritative state and interpretive guidance.

5.2 Decision-Trace Plane: What Must Be Recorded for Audit and Debugging

The decision-trace plane records what the agent did while producing a decision and, when applicable, while initiating tool-mediated actions. This record is essential for auditability because it allows an auditor to answer questions that a narrative explanation cannot answer reliably: which sources were consulted, what constraints were checked, what was executed, and what was blocked or abandoned [61].

At minimum, a decision trace should capture [61,62]:

1. Evidence references: identifiers for retrieved items (document IDs, log query hashes, metric query definitions), timestamps, and the retrieval scope and filters used.
2. Tool invocations: tool name, parameters (redacted as needed), privilege context, success/failure status, and returned outputs or summaries linked to raw outputs.

3. Intermediate checks: precondition validations, policy evaluations, safety checks, and any dry-run results.
4. Decision artifacts: the final recommendation or action proposal, uncertainty statements where applicable, and the mapping from claims to evidence identifiers.
5. Execution record when relevant: what was attempted, what succeeded partially, and what rollback or monitoring steps were triggered.

Two points are easy to overlook and should be treated analytically rather than as engineering detail: Two points are easy to overlook and should be treated analytically rather than as engineering detail. First, auditability depends on trace completeness and controlled retention. Completeness does not mean recording everything. Rather, it is a structural property of the record: the trace should contain the minimum linked artifacts needed to reconstruct the decision with evidentiary linkage and to assess whether constraints were respected.

Trace sufficiency is a separate question. It concerns whether the recorded material shows an adequate evidentiary basis for the decision under the relevant risk and time constraints. Controlled retention also matters because traces can contain sensitive operational content and can themselves become a privacy and security liability. An auditable system therefore needs explicit trace governance: what is retained, for how long, under what access controls, and how trace integrity is protected [63,64].

In practice, these instrumentation requirements also carry non-trivial computational and storage implications at scale. More extensive trace capture can increase latency, reduce throughput, and introduce operational overhead, particularly in microservice-based or distributed environments where instrumentation is applied continuously across many components. It can also generate large volumes of provenance or telemetry data, which creates storage, indexing, retention, and query-efficiency burdens that must be managed explicitly rather than treated as a negligible implementation detail. For that reason, trace completeness should not be interpreted as exhaustive logging by default, but as risk-calibrated completeness: enough structured capture to support reconstruction and audit, while remaining feasible under the performance, storage, and governance constraints of the environment [56,65,66].

Second, reproducibility requires evidence snapshots or defensible approximations. If a decision trace only stores a textual summary of retrieved evidence, reconstruction becomes dependent on the agent's summarization behavior. When possible, traceability is stronger when the system stores immutable references to evidence at the time of retrieval, or stores query definitions that can be re-run against versioned data [67]. In environments without versioned evidence stores, the review should be explicit that auditable autonomy is constrained by infrastructure maturity; this is a limitation, not a minor implementation choice [68].

5.3 Outcomes Plane: What Makes Assurance Empirically Meaningful

The outcomes plane records what happened after a decision was adopted or an action was executed. Outcomes are the empirical basis for deciding whether autonomy is beneficial, safe, and stable over time. Without outcome records, assurance collapses into claims about process rather than verified effects [69].

Outcome capture should include a mix of service-level indicators and governance indicators [70,71]:

1. Service and reliability outcomes: KPI changes, time-to-mitigation, time-to-recovery, incident recurrence, degradation persistence.
2. Rollback and recovery outcomes: rollback frequency, rollback success, collateral effects, time to stabilize after rollback.
3. Human interaction outcomes: override frequency, escalation rates, approval latency, operator disagreement patterns.

4. Security and compliance outcomes where relevant: policy violations, audit flags, unauthorized access attempts, containment effectiveness.

Fig. 1 presents the three-plane foundation for auditable autonomy, distinguishing operational evidence, agent decision traces, and post-decision outcomes, while emphasizing that the recorded links between planes capture provenance and evaluation linkage rather than causal proof.

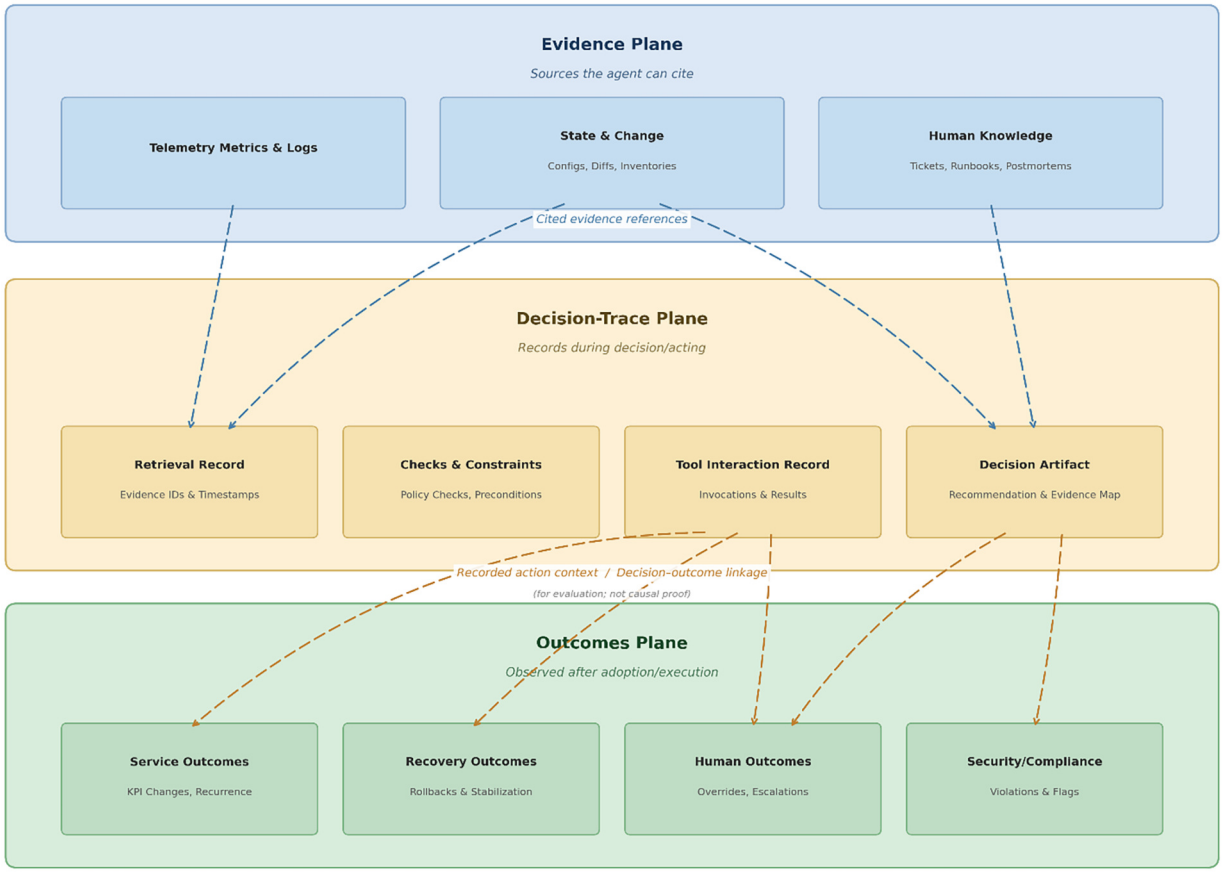

Figure 1: Three-plane model for auditable autonomy with recorded associations. Note: Association links record provenance and linkage; they do not assert causality.

However, outcomes are also the easiest plane to misread. Operations are confounded by concurrent changes, seasonality, evolving baselines, shifting workloads, and partial interventions. A post-action KPI improvement does not establish causality, and a post-action KPI deterioration does not automatically establish fault, particularly when multiple actions are taken under stress [72].

For that reason, outcome interpretation must be qualified by evaluation design, and the outcome plane must be linked to the decision-trace plane so that the sequence of checks and actions is visible. This is the practical reason the three planes should be treated as a linked evidentiary foundation rather than as separate reporting streams [73]. Table 3 summarizes the artifact classes required to support auditable autonomy, specifying for each artifact its freshness expectations, authority level, common failure modes, and the assurance mitigations used to preserve evidentiary integrity and trace completeness.

Table 3: Artifacts required for auditability and their operational implications.

Artifact Type	Freshness Requirement	Authority Level	Typical Failure Mode	Mitigation for Assurance
Time-series metrics	Near-real-time to minutes, depending on decision	Heuristic signal	Ambiguity; spurious attribution; missing instrumentation	Cross-source validation; uncertainty representation; require corroboration for prescriptions
Logs	Near-real-time to hours	Mixed; depends on coverage	Incomplete coverage; log injection; version mismatch	Version-aware parsing; integrity controls; corroborate with state/config evidence
Distributed traces	Near-real-time with sampling delays	Mixed; sampled evidence	Sampling bias; missing spans; instrumentation drift	Sampling-aware interpretation; trace completeness indicators; fall back to alternative evidence
Configuration state and histories	Minutes to hours, but must be current for execution	Often authoritative	Out-of-band changes; inconsistent sources	Require provenance; reconcile across stores; restrict autonomy when provenance is weak
Change diffs/deployment records	Minutes to hours	High for governed changes	Missing context; incomplete lineage	Link to approvals; require metadata; incorporate "change recentness" checks
Tickets/incident records	Hours to days	Contextual guidance	Inconsistent labeling; narrative bias	Normalize via schema; treat as hypotheses; validate against live state
Runbooks/ postmortems	Days to months, but must be version-aligned	Guidance, not proof	Stale procedures; environment mismatch	Version tagging; retrieval scoping; enforce precondition verification before action
Retrieval record (IDs, timestamps, scope)	Must be exact at decision time	Trace evidence	Missing IDs; non-replayable queries	Store immutable references where possible; preserve query definitions; integrity checks
Tool invocation record (tool, params, privileges, result)	Exact at execution time	Trace evidence	Partial execution; silent failures; permission denials	Record status codes and outputs; require post-action verification; rollback readiness

(Continued)

Table 3 (continued)

Artifact Type	Freshness Requirement	Authority Level	Typical Failure Mode	Mitigation for Assurance
Policy and precondition check results	Exact at decision/execution time	Assurance evidence	Checks skipped; checks not logged	Make checks mandatory; log check outcomes; block high-risk actions on missing checks
Outcome measurements (KPIs, overrides, rollbacks)	Minutes to days, depending on outcome	Evaluation evidence	Confounding; shifting baselines	Use controlled rollouts when possible; attach context; treat attribution as qualified

6 Architectures and Operational Patterns for Agentic Systems

This section discusses agentic architectures only to the extent that they expose places where decision assurance controls can be enforced. The review does not treat architecture as an end in itself. The analytic question is narrower: which operational patterns increase autonomy risk, and which design choices make assurance feasible without relying on informal operator vigilance as the primary safeguard.

Two recurring problems motivate this focus. First, many autonomy claims omit the specifics of tool permissions and the privilege model, even though the same recommendation is materially different when it can be executed through a privileged tool context [74]. Second, operational actions often fail in partial or non-obvious ways: a tool call can succeed while producing an unintended side effect, fail silently due to permissions, or only apply to a subset of resources. Any architecture that does not represent and record these realities tends to create brittle automation, even when the language output appears competent [75,76].

6.1 Agentic Patterns in Operational Workflows

Planner–executor patterns separate high-level reasoning from action-oriented tool use. The planner decomposes a task, selects evidence to consult, and proposes candidate steps; the executor carries out a subset of those steps through tools [77,78]. This separation is often presented as a safety feature, but it is only protective when the boundary is enforceable. If the executor inherits broad privileges or if the planner is allowed to generate tool instructions that the executor treats as authoritative without verification, the separation becomes cosmetic [79]. The operational benefit of this pattern is that it can localize risk: verification and gating can be attached to the executor interface, and the planner can be constrained to produce proposals that remain non-binding until checks pass.

Tool-augmented agents integrate tool calls into the reasoning process, typically mixing evidence gathering, intermediate checks, and workflow updates in a single loop. This pattern can be operationally efficient, yet it intensifies assurance requirements because errors can accumulate across steps. A common failure is implicit state drift: an early tool output is treated as valid later even when the environment changes. Another is privilege creep: adding one more tool over time expands the action surface until the agent effectively has administrative capabilities [80]. In this pattern, assurance is less about improving the model's reasoning and more about limiting what tools can do, under which conditions, and what evidence must be present before a tool can be invoked.

Retrieval-augmented agents treat retrieval as the mechanism for grounding. In operations, retrieval can pull from runbooks, tickets, prior incidents, and postmortems, sometimes alongside technical telemetry summaries. The critical issue is that retrieval does not eliminate the need for verification [81]. Retrieved guidance can be stale, context-mismatched, or simply wrong for the current configuration. An architecture that treats retrieval results as authoritative is therefore unsafe by design [82]. The more defensible posture is to treat retrieved material as candidate guidance that must be reconciled with current state and policy constraints, with explicit handling for missing or conflicting evidence.

Multi-agent decompositions distribute responsibilities across specialized agents, such as a triage agent, an evidence-collection agent, a verification agent, and an action agent. This can improve modularity and make responsibilities explicit, but it introduces coordination risk. Agents can produce conflicting recommendations, duplicate tool calls, or create oscillating actions in dynamic environments [83]. A frequent omission in published systems is a clear arbitration mechanism with traceable justification: who resolves conflicts, based on what constraints, and how that resolution is recorded [84]. Without arbitration and trace capture, multi-agent systems risk becoming less auditable than monolithic ones, even when each individual component is well designed.

Across these patterns, two requirements remain necessary for defensible autonomy: an explicit privilege model for tool use, and trace capture that records both successful and failed tool interactions. Architectural accounts that omit these requirements do not provide enough detail to support auditability or credible claims about safe autonomy.

6.2 Assurance Insertion Points as Design Constraints

Assurance controls are often described as added to an agent. In practice, they must be treated as design constraints that determine what an agent can do, when it is permitted to do it, and what must be recorded [85]. Three insertion points are consistently consequential across operational domains.

First, after evidence is assembled, assurance depends on constraining scope and provenance. The system should record what sources were consulted and under what filters, while also enforcing retrieval boundaries that reduce exposure to untrusted or irrelevant content. This is where freshness controls, source allow-lists, and context scoping are most effective, because they shape the set of candidate evidence the agent can rely on [86,87].

Second, during verification, assurance depends on explicit precondition checks and policy tests. Verification should not be framed simply as a model that checks its work. It should be framed as a set of enforceable validations that are external to the model's narrative: dry-runs, configuration validation, dependency checks, and policy conformance tests [88,89]. When verification is infeasible within the time budget, the architecture should support staged actions and stronger rollback readiness rather than skipping checks entirely [38].

Third, immediately before any privileged action, assurance depends on gating. Gating is not a prompt technique; it is a control boundary. It includes least-privilege tool scopes, approval requirements for high-impact actions, execution limits that prevent repeated or cascading actions under uncertainty, and explicit conditions under which the agent must stop and request human authorization. The key analytic point is that action gating is an architectural commitment: if a system cannot enforce it, then autonomy claims should be limited to recommendation support [90,91].

Alongside these insertion points, trace capture must be continuous. Trace capture is not a separate step; it is the instrumentation base that enables auditability and later evaluation. If traces only capture final recommendations, or if they omit failed tool calls and rejected hypotheses, the system becomes difficult

to audit and harder to improve [92]. For that reason, trace capture and retention rules should be designed together with the privilege model and gating policies.

Fig. 2 below summarizes a generic agentic operations workflow and highlights the specific control points where assurance constraints must be enforced (scope/provenance during evidence assembly, policy checks during verification, gating before privileged actions), while emphasizing trace capture as a cross-cutting requirement for audit and evaluation.

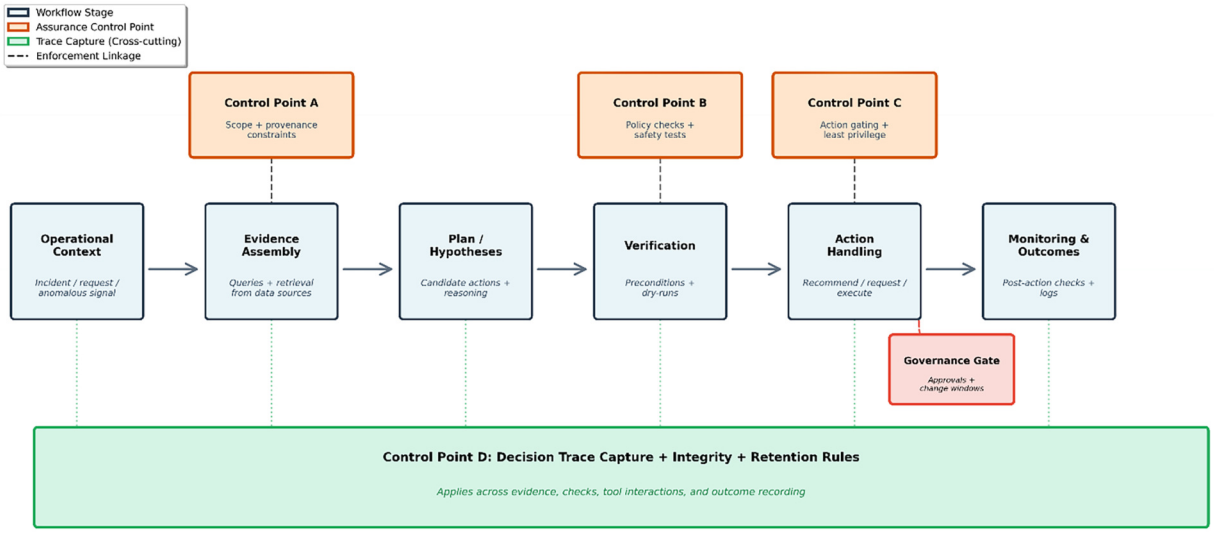

Figure 2: Agentic operational workflow schematic with assurance control points. Note: Assurance controls are shown as constraint modules; dashed links indicate where a control must be enforced. Connectors show linkage for design and audit; they do not specify a required execution order or causal claim.

7 Decision Assurance Mechanisms

This section defines the assurance stack as a set of distinct control layers that constrain how an agent forms, justifies, and operationalizes decisions. The stack is descriptive in that it classifies the control functions reported across the literature, but it is also minimally prescriptive because any defensible autonomy claim should make clear which layers are implemented, how they interact, and where residual risk remains.

The layers are not interchangeable. Each mitigates a different class of failure, draws on different artifacts, and introduces different operational costs. Treating assurance as a single capability claim obscures these differences and encourages weak evaluations that report apparent success without specifying what was actually controlled [93,94].

7.1 Assurance Stack Taxonomy

Layer A: grounding and evidence quality controls

This layer constrains what the agent is permitted to treat as support for a claim. Provenance constraints restrict acceptable evidence sources and retrieval scopes. Cross-source consistency checks prevent the agent from relying on a single ambiguous signal when independent signals conflict. Evidence sufficiency checks formalize the idea that some decisions should not be made when key signals are missing or stale, even if the model can produce a plausible narrative. A core critique is that retrieval is often treated as grounding by default. In operational settings, retrieval is only a candidate input; grounding requires explicit rules for authority, freshness, and conflict handling [95,96].

In practice, such thresholds should be validated as domain-calibrated operating conditions rather than as universal cutoffs. A defensible approach is to define required signals, freshness bounds, and abstention triggers with domain experts, then test those conditions through historical replay, missing-signal stress scenarios, or comparable deployment-relevant validation settings [97,98]. In effect, evidence-sufficiency thresholds should be treated as selective-decision thresholds: they must be calibrated against the risk of acting on insufficient evidence vs. the operational cost of abstaining, and they should be revised when the decision surface or evidence environment changes.

Layer B: verification and precondition checking

Verification shifts assurance away from narrative self-consistency and toward externally checkable conditions. Dry-runs, configuration validation, dependency checks, and policy validation are typical mechanisms. Uncertainty-aware abstention belongs here because it operationalizes a refusal condition: the system should stop, request additional evidence, or require human review when verification cannot be completed within the available time and privilege constraints [99–101]. A common gap in published systems is that verification is described as "the agent checks" rather than as a concrete set of validations that are logged, repeatable, and independent of the model's text generation.

Layer C: action gating and risk bounding

Gating is the boundary that prevents unverified recommendations from becoming unbounded actions. It includes allow-lists for action types, least-privilege tool scopes, staged automation modes that limit blast radius, and approval requirements for high-impact decisions. The analytic point is that gating is a system property, not a prompt property [102]. If the environment cannot enforce it, then the agent should be treated as a recommendation system regardless of how autonomous the interface appears [103]. Another under-discussed issue is partial execution: gating policies should treat tool failure and partial completion as first-class outcomes, not as edge cases.

Layer D: auditability and accountability controls

Auditability is the layer that makes assurance inspectable, contestable, and improvable. Trace completeness requires that the decision record includes evidence references, checks performed, tool interactions (including failures), and the final decision artifact with explicit linkage. Reproducibility requires enough stable referencing to re-create the decision context, at least approximately, without depending on a post-hoc narrative [104]. Separation of duties and governance controls ensure that the same agent identity is not simultaneously proposing, approving, and executing privileged actions [95]. A frequent weakness is that systems log the final answer but not the evidentiary and constraint path that produced it; such systems can appear explainable while remaining unauditable.

To illustrate how these layers interact in a single operational workflow, consider the following simplified incident-response scenario involving an agentic recommendation for service rollback: An agent detects a latency spike in a production service.

- Layer A: it checks whether the evidence comes from authoritative and recent telemetry, and whether logs, metrics, and recent change records are mutually consistent.
- Layer B: before proposing rollback, it runs precondition checks such as change-window status, dependency checks or dry-run validation.
- Layer C: even if rollback appears justified, the system cannot execute it freely if the action exceeds its privilege scope or requires approval because of potential blast radius.
- Layer D: the system records the evidence references, the checks performed, the gating outcome, the final recommendation, and any later override or rollback result, so the decision can be reconstructed and evaluated afterward.

Fig. 3 overlays the assurance stack onto the three-plane big data foundation introduced earlier, clarifying which control layers depend primarily on operational evidence, which require detailed decision traces, and which are only meaningful when linked to outcome records for monitoring and evaluation.

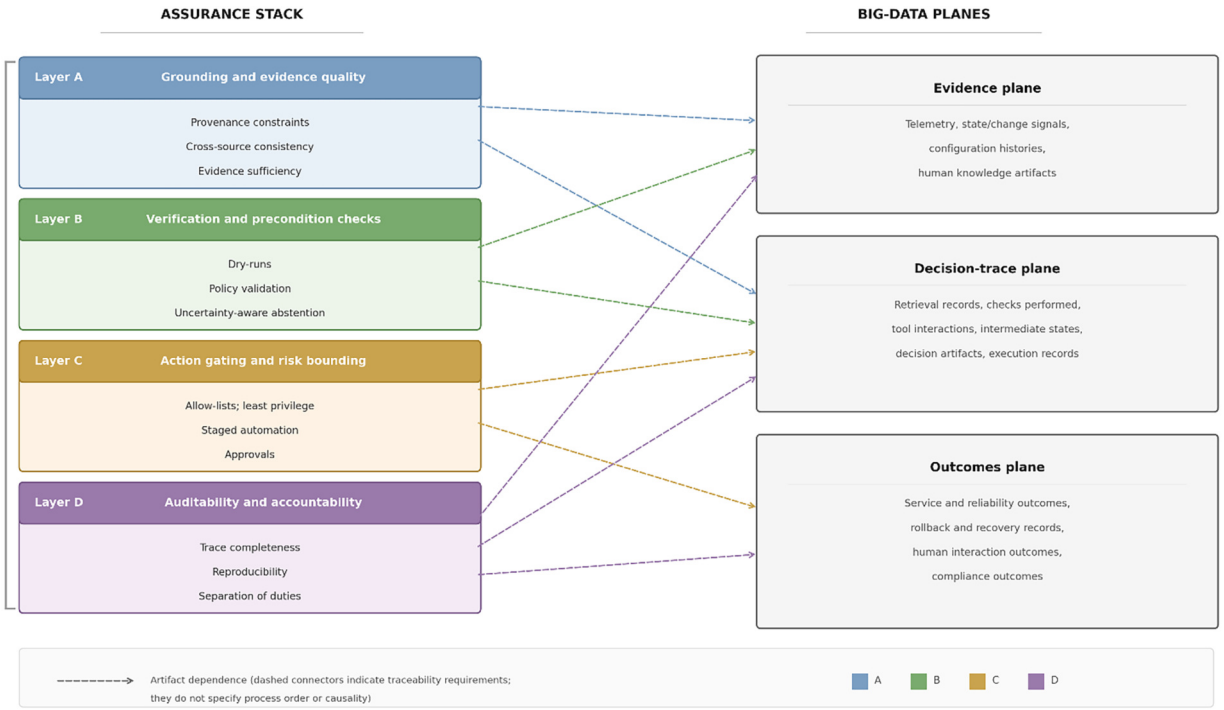

Figure 3: Assurance stack and its dependence on big data planes. Note: Dashed connectors indicate artifact dependence and traceability requirements; they do not specify process order or causality.

7.2 Trade-offs and Failure Cases

Assurance mechanisms create friction, and that friction is not uniformly undesirable. In operations, friction often functions as risk control. The critical question is whether a given layer imposes costs that are proportionate to the decision space and whether those costs are made explicit in evaluation.

When these layers are implemented together, their costs and benefits do not simply add up; they can also conflict. In resource-constrained environments, stronger evidence checks may increase latency, more extensive verification may reduce throughput, tighter gating may shift workload back to operators, and richer trace capture may increase storage, privacy, and governance burdens [56]. These tensions do not make concurrent assurance infeasible, but they do mean that layer combinations must be calibrated to the decision surface, especially with respect to reversibility, blast radius, and time sensitivity.

Layer A trade-offs and failure modes

Costs typically appear as reduced coverage and increased latency in evidence assembly, especially when provenance constraints exclude convenient but untrusted sources. Cross-source consistency checks can increase false rejections when instrumentation is incomplete or when signals naturally diverge under benign conditions. Residual risk remains when authoritative sources themselves are wrong or outdated, or when the environment is only partially observable. Failure can manifest as brittle retrieval behavior, where the system overfits to a narrow evidence set and misses relevant signals that fall outside the enforced scope.

Layer B trade-offs and failure modes

Verification can be expensive in time and compute, and it can create operational burden when checks depend on specialized tooling or manual preparation. Overly strict preconditions may also trigger frequent abstention, shifting workload back to operators without a clear net benefit. Residual risk remains when verification is superficial, when dry-runs fail to capture real side effects, or when checks are executed without being recorded in an auditable decision trace. In practice, a recurring failure mode is nominal verification: systems claim that validation occurred, but the evidence of what was checked, with what inputs, and with what outcomes is missing, non-reproducible, or no longer aligned with the current environment due to drift in configurations, dependencies, or policies.

Layer C trade-offs and failure modes

Gating reduces speed and can complicate workflow integration, especially when approvals introduce queueing delays. Least-privilege tool scopes can cause frequent tool failures that the agent must handle gracefully, and staged automation can create coordination overhead across teams. Residual risk remains when the allow-list is too broad, when privilege boundaries are misconfigured, or when repeated low-risk actions accumulate into a high-impact outcome. A typical failure pattern is action fragmentation: each individual step appears acceptable under local gating rules, yet the combined sequence creates unacceptable systemic change. Another is hidden escalation, where the agent indirectly triggers high-impact effects through a chain of safe tool calls.

Layer D trade-offs and failure modes

Trace capture, integrity controls, and retention rules introduce storage and governance costs and can create privacy and security liabilities if not managed carefully. Strong reproducibility requirements can be difficult to satisfy when data stores are not versioned or when operational evidence is ephemeral. Residual risk persists through trace incompleteness, trace tampering, or over-redaction that destroys audit usefulness. Failures often manifest as irreconcilable post-incident narratives: the agent's text claims one basis for action, but the recorded artifacts cannot confirm it, or key steps are missing and cannot be re-created.

Table 4 that follows, maps representative assurance mechanisms to the failure modes they reduce, the artifact inputs they require, the operational overhead they introduce, and the residual risks that remain even when the mechanism is correctly implemented.

Table 4: Assurance mechanisms mapping.

Assurance Mechanism	Failure Modes Reduced	Required Artifact Inputs	Operational Overhead	Residual Risk and Typical Limitations
Provenance constraints (source allow-lists, scoped retrieval)	Use of untrusted guidance; prompt-injection via uncontrolled corpora; irrelevant evidence	Evidence references with source metadata; retrieval scope logs	Reduced coverage; increased retrieval engineering; potential latency	Authoritative sources can still be wrong; overly narrow scope can hide relevant signals

(Continued)

Table 4 (continued)

Assurance Mechanism	Failure Modes Reduced	Required Artifact Inputs	Operational Overhead	Residual Risk and Typical Limitations
Cross-source consistency checks	Single-signal overcommitment; spurious attribution; contradiction blindness	At least two independent evidence streams; decision trace capturing comparisons	Additional queries; reconciliation logic; increased abstentions	Correlated evidence sources can fail together; inconsistent instrumentation can generate false contradictions
Evidence sufficiency thresholds (required signals, freshness bounds)	Confident decisions under missing/stale evidence	Evidence freshness metadata; missing-signal indicators; trace records of sufficiency evaluation	More "stop and ask" events; potential operator frustration	Threshold tuning is context-dependent; attackers can target required signals to force abstention
Dry-runs and change simulation	Unsafe changes due to untested assumptions; hidden dependencies	Tool invocation records; simulation results; configuration snapshots	Tooling complexity; added time per action	Simulations often omit real side effects; dry-run success does not ensure safe execution
Policy validation (compliance, change windows, separation rules)	Governance violations; unauthorized actions; policy drift	Policy artifacts; policy evaluation logs; privilege context	Maintenance burden; possible false blocks during emergencies	Policies can be incomplete or outdated; emergency exceptions can become informal backdoors
Uncertainty-aware abstention with escalation	Overconfident prescriptions; risky automation under ambiguity	Confidence/uncertainty signals; evidence sufficiency outcomes; escalation logs	Increased human workload; longer resolution time	Poor calibration; strategic abstention can hide low competence; escalation pathways can be gamed

(Continued)

Table 4 (continued)

Assurance Mechanism	Failure Modes Reduced	Required Artifact Inputs	Operational Overhead	Residual Risk and Typical Limitations
Allow-lists for action types and parameters	Tool misuse; high-impact actions by default; arbitrary action selection	Action taxonomy; parameter constraints; trace of gating decisions	Reduced flexibility; continuous tuning as tool set evolves	Allow-lists can be too permissive; sequences of "safe" actions can still produce unsafe outcomes
Least-privilege tool scopes	Lateral movement; broad blast radius from a compromised agent	Privilege model; tool access logs; denied-action traces	Integration complexity; more tool failures to handle	Misconfigured roles; privilege creep over time; indirect effects through shared tooling
Staged automation modes (recommend-only, supervised, limited autonomy)	Unbounded execution; premature remediation	Mode metadata; approval records; post-action verification logs	Process overhead; slower automation benefits	Mode escalation criteria may be unclear; staged steps can still accumulate into high-impact change
Trace completeness requirements (evidence IDs, checks, tool calls incl. failures)	Unauditable decisions; post-incident ambiguity; hidden tool errors	Decision-trace plane artifacts; integrity metadata	Storage and governance cost; privacy considerations	Over-redaction; missing links; trace tampering or loss
Reproducibility controls (versioned references, query definitions)	Non-repeatable evaluations; unverifiable claims	Evidence snapshots or version pointers; query logs; environment metadata	Engineering maturity requirements; higher storage	Many environments cannot fully version evidence; replay may still diverge under drift
Separation of duties and approval workflows	Self-approval; unsafe concentration of authority	Identity and role logs; approval artifacts; governance rules	Latency; organizational coordination	Social engineering; approval rubber-stamping; emergency bypass mechanisms

8 Evaluation beyond Text

This section specifies how agentic operational systems should be evaluated when model outputs can shape or trigger consequential actions. The core premise is that language quality is, at best, an entry condition. Claims about autonomy require evidence that decisions were supported by traceable artifacts, that actions respected constraints, and that observed outcomes are interpreted with appropriate caution about confounding and drift.

8.1 Evaluation Designs

Offline replay over historical corpora is often the most practical starting point because it allows controlled experimentation against known incident timelines and known operational states. Replay designs should be explicit about what is frozen (evidence snapshots, configurations, ticket states) and what is re-derived at evaluation time (queries re-run against a versioned store, or approximated by stored retrieval results) [105]. A major validity threat is replay leakage: if future information is inadvertently exposed through postmortems, resolved tickets, or labels created after the decision point, the evaluation will overstate competence [106,107].

Stress testing and counterfactual evaluation are needed because operational failure tends to concentrate in edge cases: ambiguous evidence, partial outages, degraded telemetry, and inconsistent runbook guidance [108]. These tests should deliberately introduce missing-signal conditions, contradictory evidence, degraded tool availability, and adversarial or untrusted artifacts in evidence stores [41,109]. The goal is not to inflate failure rates, but to characterize failure modes that are operationally plausible and disproportionately costly [110].

Controlled rollout designs are the main route to credible outcome claims [111]. A defensible rollout separates decision modes (recommend-only vs. supervised execution vs. tightly bounded autonomy) and states the gating policy and privilege scope in advance [112]. In addition, post-incident audits should be integrated into the evaluation protocol, not treated as an optional narrative. Audits assess whether decisions can be reconstructed from recorded traces, whether checks were actually performed, and whether the system's behavior is consistent with governance constraints [73].

Recent literature also offers more concrete illustrations of what evaluation beyond text can look like in practice. For example, benchmark work on stateful tool use has shown that many agent evaluations remain too narrow when they focus on single-turn or partial tool interactions rather than longer-horizon, multi-step execution sequences [113]. In a more policy-constrained operational setting, JourneyBench evaluates customer-support agents not only on task completion but also on adherence to multi-step business rules and workflow dependencies [114]. Related domain-specific studies in business process modeling and software engineering likewise demonstrate the value of evaluation designs that move beyond surface response quality toward workflow validity, structural correctness, and operational utility [115,116]. These examples do not eliminate the need for stronger cross-domain standards, but they show that more decision-relevant and action-aware evaluation designs are already feasible in adjacent literatures.

8.2 Metrics Taxonomy

Metrics should be selected to match the evaluation level being claimed, and they should be instrumentable through the linked record developed earlier. Reporting only aggregate success rates is rarely informative because operational decisions differ sharply in reversibility, blast radius, and verification feasibility. Where aggregation is necessary, studies should stratify by decision intent and action surface, or provide consequence-weighted summaries [117–119]. Fig. 4 presents an evaluation ladder-type framework for agentic

operations in which higher levels require progressively stronger evidence—moving from response quality to evidence fidelity, action validity, and finally longitudinal outcomes—while making explicit the reporting artifacts needed to interpret results.

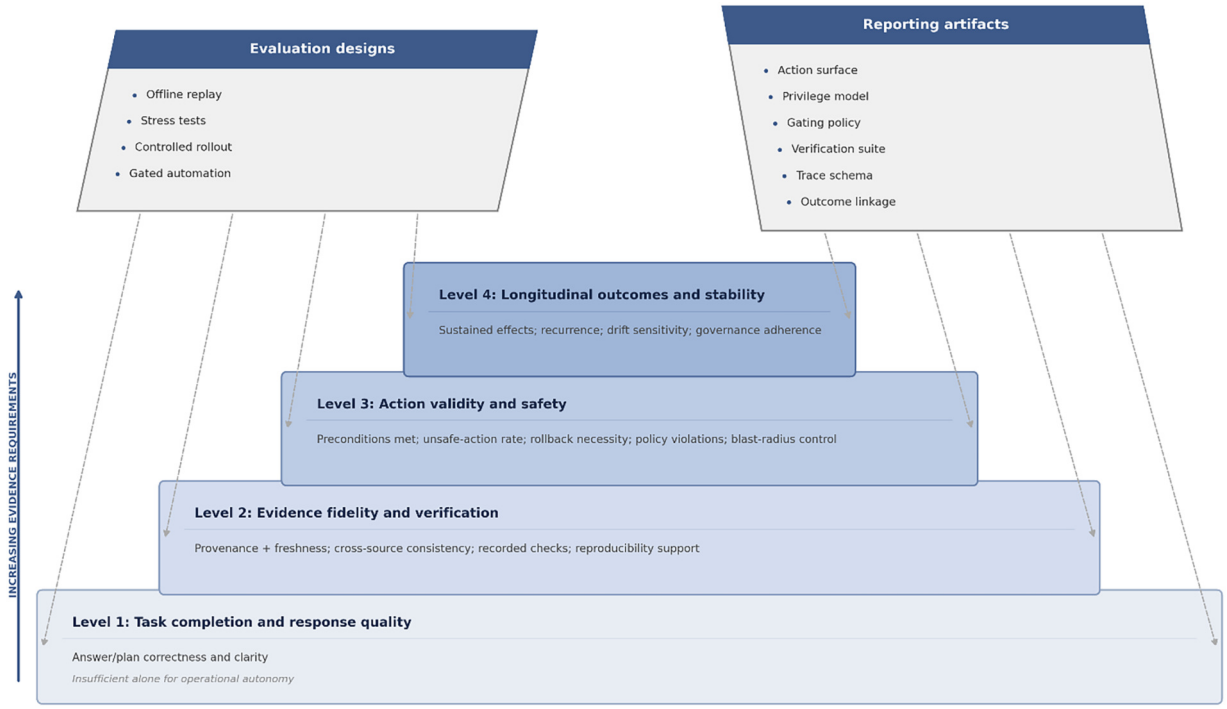

Figure 4: Evaluation ladder for agentic operations (nested levels).

The conceptual ladder depicted in Fig. 4 is intended as a progression in evidentiary rigor rather than as a fixed scoring rubric. Advancement between levels should therefore be interpreted through minimum validation conditions, not universal numeric thresholds. A study should not claim evidence-fidelity evaluation unless substantive claims are linked to identifiable evidence sources with source and time metadata. It should not claim action-validity evaluation unless required checks, policy constraints, and action outcomes are recorded in a reconstructable trace. It should not claim longitudinal-outcome evaluation unless decision traces can be linked to post-action records over time and the treatment of confounding or concurrent interventions is stated explicitly. The appropriate strictness of these conditions may vary by decision surface but the reporting logic should remain explicit so that the claimed evaluation level is reproducible and comparable across studies.

In domains with limited historical incident data or heavily confounded outcome environments, the ladder should still be applied progressively but studies should stop at the highest level that can be supported credibly; in such cases, evidence-fidelity and action-validity evaluations may be defensible even when robust longitudinal-outcome claims are not.

Table 5 below summarizes evaluation metrics that map directly to decision assurance, specifying what each metric measures, which artifacts are required to compute it, and the most common confounders that can distort interpretation in operational settings.

Table 5: Metrics catalog for agentic operational evaluation.

Metric Family	What It Measures	Instrumentation Requirements	Common Confounders/Threats to Validity
Evidence provenance coverage	Fraction of substantive claims linked to evidence identifiers with source and timestamp	Retrieval record with stable IDs; evidence metadata (source, time, scope)	Evidence exists but is not authoritative; missing metadata; over-linking to irrelevant artifacts
Evidence freshness compliance	Whether cited evidence meets freshness bounds appropriate to the decision surface	Freshness rules; timestamps; decision time markers	Clock skew; delayed ingestion; stale but still "correct" guidance masking risk
Cross-source consistency rate	How often key claims are corroborated (or contradicted) by independent sources	Evidence-plane diversity indicators; recorded comparison outcomes	Correlated evidence sources failing together; inconsistent instrumentation causing false contradictions
Check execution completeness	Whether required verification checks were executed and recorded	Trace logs for check suite; check inputs/outputs; pass/fail states	Checks executed but not logged; version mismatch between checks and current system; shallow checks that miss side effects
Abstention and escalation appropriateness	Whether abstentions occur when evidence is insufficient or verification is infeasible, and whether escalations follow policy	Sufficiency outcomes; abstention triggers; escalation routing artifacts	Overly conservative abstention masking low competence; inconsistent human follow-through after escalation
Unsafe-action rate	Rate of actions that violate policy, exceed allowed scope, or occur without required checks	Tool invocation records; policy evaluation logs; privilege context	Hidden actions via indirect tool chains; mis-specified policies; partial tool failures not captured
Blast-radius adherence	Whether executed actions stay within bounded scope (resources, services, time windows)	Action scope metadata; asset inventories; change boundaries	Incomplete inventory; shared dependencies expanding true radius; scope definitions that are too coarse
Rollback incidence and success	Frequency of rollback and whether rollback restores acceptable state without collateral effects	Outcome logs; rollback triggers; stabilization markers	Concurrent interventions; delayed effects; rollbacks that "fix the metric" while introducing latent debt

(Continued)

Table 5 (continued)

Metric Family	What It Measures	Instrumentation Requirements	Common Confounders/Threats to Validity
Operator override rate with reasons	How often humans reject or modify agent outputs, and why	Override events; annotations; approval logs	Documentation fatigue; overrides performed without recorded rationale; organizational norms influencing overrides
Time-to-mitigation/time-to-recovery (stratified)	Operational responsiveness and recovery performance under defined modes	Incident timeline markers; mode metadata (recommend/supervised/limited autonomy)	Severity mix changes; seasonality; parallel human efforts not accounted for
Recurrence and regression rate	Whether similar incidents recur after agent-guided actions, and whether performance degrades across time	Outcome history; incident similarity criteria; trace linkage	Drift in workloads; changing system architecture; evolving incident definitions
Audit reconstructability score	Whether an independent reviewer can reconstruct what was checked, what evidence was used, and what action occurred	Trace completeness; integrity metadata; access-controlled retention	Over-redaction; missing links; retention limits; inability to replay due to non-versioned evidence stores

8.3 Reporting Requirements for Comparable Results

To make evaluation claims interpretable across studies, reporting should include: the action surface (what the system can influence), the privilege model (what it is allowed to execute), the gating policy (what is blocked or requires approval), the verification suite (what checks exist and how outcomes are logged), a trace schema summary (what is recorded and retained), and a clear statement about how outcome linkage is treated (association for evaluation rather than causal proof unless a stronger design is used).

9 Failure Modes, Security, and Adversarial Considerations

This section limits overclaiming by making failure explicit and by tying failures back to the assurance stack and to trace evidence. Two boundary conditions guide the analysis. First, many high-impact failures in agentic operations are not best understood as hallucinations. They arise from mis-specified decision surfaces, weak privilege boundaries, incomplete instrumentation, or brittle verification practices [120]. Second, security threats cannot be treated as an external add-on. Tool access, evidence retrieval, trace collection, and governance controls expand the attack surface, and the same mechanisms that enable assurance can be targeted for audit evasion or trace corruption [121,122].

9.1 Accidental Failure Modes

Stale or context-mismatched evidence

Evidence can be stale because ingestion is delayed, because the evidence source itself is outdated, or because retrieval pulls the wrong version of runbooks and postmortems. In operational settings, stale-but-plausible logic is particularly dangerous: it can justify actions that were correct for a prior architecture and harmful for the current one. This failure is often not a model error in the narrow sense; it is an evidence governance failure and a retrieval scoping failure (Layer A), sometimes compounded by a missing check suite that would catch environment mismatch (Layer B). Decision traces should therefore record evidence timestamps, version identifiers where available, and retrieval scope constraints so auditors can assess whether staleness was foreseeable at decision time [61,123].

Ambiguous and under-determined signals

Telemetry frequently supports multiple hypotheses. When systems are partially observable, ambiguity is normal, not exceptional. A recurrent failure is forced commitment: the agent produces a single best explanation response and proceeds as if uncertainty is resolved. This is a design and evaluation failure as much as a modeling failure. It reflects missing evidence sufficiency thresholds (Layer A) and weak abstention policies (Layer B), and it is often incentivized by evaluation protocols that reward confident resolution narratives rather than calibrated uncertainty. Trace signals that matter include explicit records of missing signals, contradiction checks performed, and whether abstention conditions were evaluated [124,125].

Tool errors and partial execution

Operational tools fail in ways that are not always visible through natural language summaries. Permissions can deny actions silently, timeouts can produce unknown partial effects, and retries can duplicate side effects. A failure mode that appears as "the agent took the action but nothing improved" may instead be "the action never executed as intended," or executed partially and created downstream instability. This is primarily a trace integrity and completeness problem (Layer D) coupled with gating mis-specification (Layer C): without detailed tool invocation records, status codes, and post-action verification, it is difficult to determine whether the decision was wrong or the execution failed. Systems that do not treat partial execution as a first-class outcome become difficult to audit reliably [8,18,126].

Recent published studies suggest that these concerns are not merely hypothetical. In implemented tool-using agents, Zhang et al. [127] showed that malfunction-amplification attacks can steer agents into repetitive or irrelevant action loops and sharply increase task failure rates, including in deployable Gmail- and CSV-based agents. In a different but complementary workflow setting, Balaji et al. [114] showed that agents in realistic customer-support scenarios may appear successful at the task level while still violating required multi-step policies, dependencies, or tool-mediated workflow conditions. Taken together, these studies support the need to treat partial execution, workflow non-compliance, and post-action verification as first-class evaluation concerns rather than edge cases.

Drift, non-stationarity, and feedback loops

Operational environments evolve: software versions change, dependencies shift, instrumentation is modified, and workload baselines move. Drift creates two practical risks. The first is that verification checks become misaligned with reality, passing even when they are no longer measuring the intended preconditions. The second is feedback: if agent recommendations influence what is logged, what operators do, or how tickets are written, the evidence base can shift in a way that makes future decisions appear better in evaluation while degrading real robustness. These issues implicate all assurance layers, but they become visible only through longitudinal outcome analysis and stable trace schemas. A minimal safeguard is to include drift indicators

and model/environment version metadata in traces, and to treat changes in observability coverage as an evaluation variable rather than noise [128–130].

Mis-specification and under-instrumentation as dominant causes

Across these accidental failures, a consistent observation is that agents often fail because the operational decision problem is ill-specified. Examples include undefined action boundaries, missing policies for what counts as sufficient evidence, and absent mechanisms for recording why an action was permitted. Under-instrumentation then makes these failures hard to diagnose. When an agent fails under these conditions, it is misleading to attribute the failure primarily to the language model; the dominant cause is usually that assurance mechanisms were never made enforceable [118].

9.2 Adversarial Manipulation

The adversarial discussion that follows can be read through a compact threat-modeling lens. The main attack surfaces are retrieved artifacts and knowledge stores, telemetry and evidence streams, tool interfaces and credentials, and the trace and governance infrastructure used for auditing and control. The attacker's objectives may include unsafe action induction, degradation of decision quality, privilege misuse, or audit evasion. The defender is assumed to control source-scoping rules, verification checks, gating policies, privilege boundaries, and trace retention or access controls, but not necessarily all upstream artifacts or telemetry inputs. The remainder of this section maps representative threats to these attack surfaces and to the assurance layers they primarily stress. Fig. 5 provides a compact threat-modeling view of the adversarial landscape discussed in this section by summarizing the main attack surfaces, attacker objectives, defender assumptions, and the assurance layers they primarily stress.

Prompt injection through untrusted artifacts

In agentic operations, prompts are not only user messages. They also include ticket descriptions, runbook text, log lines, postmortem excerpts, and other retrieved artifacts that can contain adversarial instructions or misleading content [27,131,132]. If retrieved artifacts are treated as instructions, rather than as evidence or guidance subject to constraints, the system can be manipulated into proposing or initiating unsafe tool actions. Mitigations align naturally to Layer A and Layer C: provenance constraints that restrict untrusted sources, content handling policies that prevent artifacts from being interpreted as executable instructions, and hard gating that blocks privileged actions unless verification conditions are met and logged [133].

Poisoning of runbooks and knowledge stores

Operational knowledge bases are attractive targets because they influence a wide range of decisions. Poisoning can be explicit (inserting malicious steps) or subtle (shaping prioritization guidance, normalizing unsafe practices) [134,135]. The most important point is that poisoning is not fully prevented by retrieval quality improvements. Defenses require governance around knowledge sources, versioning, review workflows, and traceability that records exactly which knowledge artifacts influenced a decision [131,136]. Verification mechanisms (Layer B) can reduce impact by requiring precondition checks and simulation before action, but they do not eliminate the risk when the checks themselves rely on poisoned guidance.

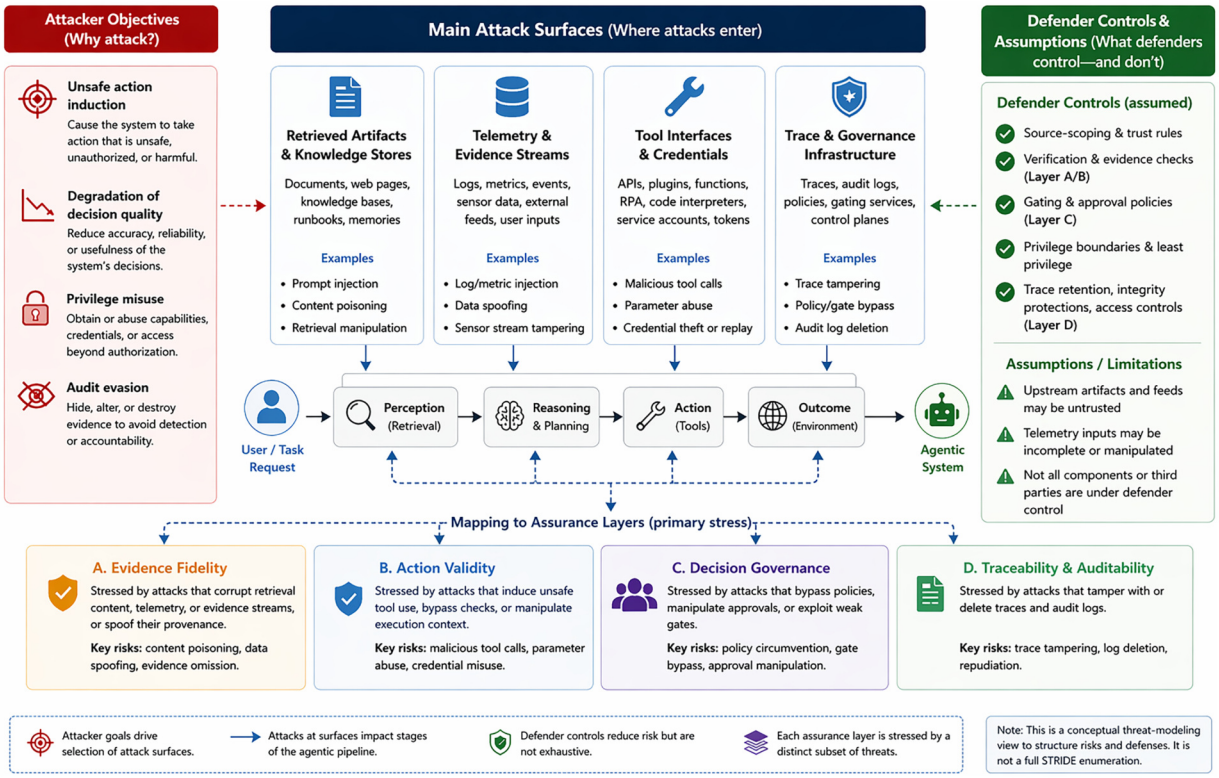

Figure 5: Compact threat-modeling view of adversarial risks, attack surfaces, defender assumptions, and assurance-layer stress points in agentic decision systems. Note: The figure is conceptual. It is intended to clarify attack surfaces, attacker objectives, defender assumptions, and their alignment with assurance layers; it is not a full STRIDE enumeration or a formal attack tree.

Log injection and telemetry manipulation

Attackers can inject log lines, distort metrics, or induce trace patterns to steer diagnosis and remediation. This is particularly relevant when the agent uses heuristic telemetry patterns to justify action [137]. Cross-source consistency checks (Layer A) are useful but not sufficient if multiple telemetry streams can be manipulated together. Trace-based detection is therefore important: decision traces should record which signals were treated as decisive, and anomaly detection should consider abrupt shifts in evidence structure, not only evidence values (for example, sudden new log templates, changes in trace completeness, or unusual correlations between independent indicators) [138,139].

Credential abuse and tool misuse

If the agent operates with broad credentials or if tool scopes are not tightly controlled, compromise can translate into high-impact actions [133]. Least privilege and explicit allow-lists (Layer C) are necessary, but they also increase reliance on correct role configuration and on clear modeling of what a tool call can do indirectly through shared infrastructure. Trace integrity controls (Layer D) become central here: without strong records of privilege context and tool invocations, it becomes difficult to distinguish accidental misuse from malicious abuse [21].

New attack surfaces created by assurance mechanisms

Assurance controls and trace collection introduce targets [140]. Attackers can seek to tamper with traces, exploit over-redaction to hide malicious steps, or flood logs and approval workflows to induce operator

fatigue and bypass [141]. Audit evasion should therefore be treated as a serious operational risk rather than as a purely hypothetical concern. A defensible position is that assurance systems must treat trace integrity, access control, and governance workflow security as part of the threat model. This is where separation of duties (Layer D) and governance gates (Layer C) matter most: they create independent checkpoints that reduce the chance that a compromised component can both act and erase evidence of acting [142].

Related empirical evidence from the audit-log and provenance literature also shows that trace-integrity concerns are operationally real rather than purely speculative. Yagemann et al. [143], for example, demonstrated that attackers can manipulate audit-log partitioning in ways that sever causal links and frame innocent users, and evaluated the problem and its mitigation across 14 real-world programs. While this work does not study LLM agents directly, it provides a strong adjacent case for treating trace integrity as a substantive security requirement whenever operational decisions depend on reconstructable execution records.

Table 6 consolidates accidental failures and adversarial threats into a single mapping that identifies the trace signals needed for detection and aligns mitigations to the relevant assurance layers, making the threat model operational rather than purely descriptive.

Table 6: Failure and threat mapping to trace signals and mitigations.

Failure/Threat Type	Trace-Based Detection Signals	Primary Mitigations (Assurance Stack Alignment)
Stale evidence or version mismatch	Evidence timestamps outside freshness bounds; missing version tags; retrieval scope too broad	Provenance and freshness constraints; evidence sufficiency thresholds (Layer A); environment-alignment checks (Layer B)
Ambiguous signals and forced commitment	Single-source dependence; absence of contradiction checks; missing-signal indicators ignored	Cross-source consistency checks; explicit abstention triggers; require uncertainty representation for diagnosis (Layer A, Layer B)
Tool errors and partial execution	Tool call failures without recorded status; repeated retries; outcome change without matching action record	Tool invocation logging with status and outputs; post-action verification; staged automation (Layer D, Layer B, Layer C)
Drift and non-stationarity	Shifts in telemetry coverage; check suite pass rates change abruptly; performance degrades over time	Versioned checks and policies; drift indicators in traces; periodic audit and recalibration (Layer D, Layer B)
Prompt injection via retrieved artifacts	Retrieved text contains instruction-like patterns; unusual tool calls follow retrieval of untrusted sources	Strict source scoping; treat retrieved text as non-executable; action gating with verification prerequisites (Layer A, Layer C)
Knowledge base/runbook poisoning	Sudden changes in runbook content; decisions cite newly edited artifacts; repeated unsafe prescriptions	Governance for knowledge edits; versioning and review; trace linkage to specific artifact versions (Layer D, Layer A)

(Continued)

Table 6 (continued)

Failure/Threat Type	Trace-Based Detection Signals	Primary Mitigations (Assurance Stack Alignment)
Log injection/telemetry manipulation	Novel log templates; abnormal correlations; inconsistent evidence across planes	Cross-source checks; anomaly detection on evidence structure; require authoritative state validation for high-impact actions (Layer A, Layer B)
Credential abuse/privilege escalation	Tool invocations outside normal scope; privilege context changes; actions exceed allow-lists	Least privilege; allow-lists; separation of duties; immutable audit logs for privilege context (Layer C, Layer D)
Trace tampering/audit evasion	Gaps in trace continuity; integrity checks fail; unusually heavy redaction; missing evidence references	Integrity protections; access controls; independent audit stores; governance review of redaction policies (Layer D)
Approval workflow fatigue/bypass	High volume of approval requests; short approval times; repeated overrides without reasons	Rate limiting; escalation rules; require rationale capture; staged autonomy policies (Layer C, Layer D)

10 Cross-Domain Synthesis and Implications

This section consolidates what generalizes across operational domains and what remains domain-specific. The objective is to avoid two common errors in the literature: implying that assurance can be solved once and then transferred unchanged, or treating every deployment setting as so unique that no transferable design constraints exist.

10.1 What Generalizes across Domains

Several requirements remain stable across operational settings because they are tied to the basic structure of agentic operations: decisions are made under partial observability, actions can have non-linear consequences, and post-incident accountability depends on reconstructable traces [144].

Evidence grounding is a baseline expectation, but it should be interpreted narrowly. The requirement is not that the agent retrieves something, but that claims are linked to identifiable artifacts with provenance and freshness constraints that match the decision surface. Without this, the system cannot be audited, and it cannot be meaningfully evaluated beyond anecdotal correctness [145,146].

Action gating generalizes even more strongly. Once tools are available, the relevant question becomes which actions are allowed under which conditions and privilege scopes. If gating cannot be enforced, autonomy claims should be restricted to recommendation-only modes regardless of interface design. This is not a conservative bias; it is the smallest defensible position when the action surface is real [146].

Auditability requirements generalize because they are the condition for learning and governance. In operational environments, failures are inevitable. The question is whether failure produces an inspectable record that supports correction and accountability [147]. Systems that do not record failed tool calls, skipped checks, or evidence identifiers are difficult to debug, and they tend to produce recurring failures that are misattributed to model unpredictability [148,149].

10.2 What Is Domain-Specific

Domain specificity is driven less by industry and more by decision properties: reversibility, blast radius, time sensitivity, and regulatory posture.

IT/SRE (reliability and service operations)

In IT/SRE settings, domain specificity is shaped by high change frequency, multi-service dependency graphs, and the practical reality that many fixes are safe only under narrow preconditions [150]. Verification often has to balance time pressure against the cost of acting on ambiguous telemetry, which makes staged autonomy and post-action validation particularly important. A recurring challenge is that operational evidence is abundant but not uniformly trustworthy: metrics and logs can be noisy, traces can be sampled, and runbooks may lag behind architecture changes [151]. As a result, assurance hinges on clear evidence authority rules, explicit handling of missing signals, and trace records that capture what checks were performed before any change was suggested or executed [152,153].

SecOps (security operations and incident response)

Security operations typically demand stricter assurance because adversaries adapt, and errors can translate into persistent exposure rather than transient degradation. Evidence sources are also more adversarial: artifacts may be crafted to mislead analysis, and prompt injection via tickets, logs, or knowledge bases is a realistic threat [154]. This domain therefore places heavier weight on provenance constraints, least-privilege tool scopes, and verifiable precondition checks that are independent of narrative reasoning [155]. Trace integrity becomes a central requirement because audit evasion and trace tampering are plausible; the system must preserve a defensible record of what was accessed, which credentials were used, and which constraints were enforced, especially when actions affect access control, containment, or credential rotation [156].

DataOps (data pipeline operations and analytics reliability)

Data operations introduces domain-specific risk through silent propagation and delayed detection [157]: a flawed change can degrade downstream models, dashboards, or reporting long before it triggers an obvious incident. The advantage of this domain is that it often has mature lineage concepts—dataset versions, pipeline DAGs, and reproducible job definitions—so auditability and replay can be more feasible than in other settings when the infrastructure is properly instrumented [158]. The downside is that autonomy claims can be overstated when agents operate on partial lineage or when decisions cite data quality signals without strong provenance. Assurance must therefore emphasize versioned evidence references, explicit definitions of correctness (schema, distributional drift, freshness), and outcome evaluation that can detect delayed effects, not only immediate pipeline success [159].

Industrial/OT (cyber-physical and process operations)

In industrial and OT contexts, domain specificity is driven by safety constraints, irreversible physical effects, and strict bounds on when interventions are permissible [160,161]. Even when an action is theoretically reversible, the cost of rollback can be high, and transient instability can create safety hazards or production losses. Verification must therefore be stronger and more conservative, often requiring simulation, interlock checks, and explicit approvals that reflect operational safety cases rather than generic policy validation. Auditability is also regulated and process-driven in many settings, so trace capture must align with existing control logs, maintenance records, and incident reporting practices, while also ensuring that the agent's role in any decision is explicitly documented [162].

Customer Ops (customer-facing operations and support workflows)

Customer operations can carry material operational and organizational risk even when the system does not directly modify production services or infrastructure. Misrouting, incorrect escalation decisions, inconsistent policy application, and mishandling of sensitive records can create legal exposure and reputational harm, with harms distributed across many small decisions rather than a single incident [163]. The domain-specific assurance burden often lies in governance rather than execution: policy consistency, access controls for personal data, retention rules, and traceability that supports dispute resolution [11]. Evaluation should therefore emphasize audit being able to reconstruct itself, override patterns, and policy violation rates, not only response quality, while remaining cautious about outcome attribution because customer satisfaction and resolution time are heavily confounded by human process variation [164].

10.3 Practical Design Implications

Three implications follow from the synthesis, stated as constraints rather than prescriptive best practices.

First, claims about autonomy should be conditioned on the action surface and privilege model, not on task labels. A system that produces correct remediation text but cannot demonstrate enforceable gating and trace completeness should be evaluated and deployed as decision support rather than as autonomous execution [165,166].

Second, assurance mechanisms should be reported as part of the experimental condition [167]. When studies omit provenance constraints, verification suites, and gating rules, their results become difficult to compare and easy to over-interpret. The absence of these details is itself evidence that the autonomy claim is not fully supported [168].

Third, model scale deserves a more explicit qualification in this context. Smaller models and frontier-scale models do not create different assurance principles, but they do change the operational trade-offs under which those principles are implemented. Smaller models may be preferable when latency, cost, local deployment or data-governance constraints dominate, especially for narrowly bounded decision surfaces with strong retrieval, fixed tool schemas, and conservative gating. In edge and mobile settings, these constraints can also limit the feasibility of repeated cross-source checks, remote verification calls, and high-volume trace capture, while privacy-sensitive deployments may require stronger controls over which evidence, traces, or outcome records can leave the device or site. Under such conditions, assurance may need to rely more heavily on compact local verification, selective trace retention, conservative action surfaces, and stronger ex ante gating rather than assuming cloud-scale observability or unrestricted audit capture. Frontier models may perform better when evidence is heterogeneous, workflows are less standardized or longer multi-step reasoning is required, but they also increase dependence on external infrastructure, cost variability, and the difficulty of reproducing behavior across settings. For that reason, model scale should not be treated as a proxy for autonomy readiness. The defensibility of an autonomy claim still depends on evidence grounding, verification, action constraints and trace completeness, not simply on model size [169–171].

Fourth, auditability should be designed to support disagreement, not only to document success [172]. A useful audit record is one that allows operators to contest a decision, identify which constraints failed, and revise policies or verification checks without relying on anecdotal recollection. In practice, this requires trace schemas that are queryable, integrity-protected, and aligned with governance workflows [140].

Table 7 summarizes cross-domain assurance expectations, distinguishing controls that are baseline requirements for defensible autonomy from those whose strictness is conditioned by reversibility, blast radius, and time sensitivity in each operational domain.

Table 7: Cross-domain assurance expectations.

Assurance Element	IT/SRE	SecOps	DataOps	Industrial/OT	Customer Ops
Layer A: Evidence grounding and quality controls	Required	Required	Required	Required	Required
Layer B: Verification and precondition checks	Context-dependent	Required	Required	Required	Context-dependent
Layer C: Action gating and risk bounding	Required	Required	Required	Required	Required
Layer D: Auditability and accountability controls	Required	Required	Required	Required	Required
Outcome linkage rigor for effectiveness claims	Context-dependent	Context-dependent	Required	Context-dependent	Context-dependent

Note: "Required" indicates a baseline expectation for defensible autonomy. "Context-dependent" does not mean optional; it indicates that the strictness of the control varies with the reversibility, blast radius, time sensitivity, and governance exposure of the decision surface in the given domain. Stricter controls should be treated as mandatory when decisions are difficult to reverse, can affect shared systems or sensitive records, operate under limited observability or high time pressure, or carry meaningful regulatory, compliance, or access-control implications.

11 Research Gaps and Future Agenda

This section identifies research gaps that are directly implied by the assurance stack and the three-plane record developed earlier. The intent is not to list generic future work, but to specify what is missing if the field wants defensible, comparable claims about agentic operational autonomy.

11.1 Standardized, Action-Level Evaluation Corpora and Replay Protocols

Current benchmarks still over-represent text-centric tasks and under-represent decisions with an explicit action surface, privilege constraints, and verifiable preconditions [54]. A priority gap is the absence of standardized replay protocols that preserve temporal realism: evidence as it existed at decision time, tool availability as it existed at the time, and constraints as they would have been enforced in practice. Without such protocols, reported performance is often inflated by inadvertent leakage through post-resolution artifacts, hindsight labeling, or access to stabilized configurations that were not available during the incident [173]. Progress here requires shared datasets that include evidence identifiers, time bounds, and a defined set of allowable tool interactions, along with reporting standards that make leakage detection feasible.

At a minimum, such a comparative benchmark suite should include: (i) temporally faithful replay tasks with frozen evidence and tool-state boundaries, (ii) verification-aware tasks in which required checks, abstentions, and gating outcomes can be assessed explicitly, (iii) partial-execution or tool-failure scenarios that test post-action validation and rollback readiness, and (iv) audit-reconstructability tasks in which an independent reviewer can determine what evidence, checks, and action constraints shaped the final decision. The point is not to impose a single universal dataset, but to define a common minimum structure that allows assurance claims to be compared across research groups without collapsing evaluation back into text-centric success rates.

11.2 Causal Attribution and Outcome Linkage under Concurrent Interventions

outcome-based claims remain hard to defend because operational environments are confounded by parallel human actions, overlapping changes, and shifting baselines [174]. Yet the field often reports improvements in recovery time or stability without a design that separates agent contribution from concurrent interventions. The gap is methodological: we need evaluation designs that explicitly model partial attribution and specify when outcome linkage is treated as association for auditing vs. when stronger causal claims are justified. This includes better use of stepped rollouts, gated autonomy with clear mode boundaries, and counterfactual evaluation approaches that leverage trace evidence to compare what was done against plausible alternative actions, while remaining explicit about validity limits [175].

11.3 Multi-Agent Coordination, Conflict Resolution, and Auditable Arbitration

Multi-agent decompositions are frequently proposed to increase reliability and modularity, but coordination failures are under-measured and under-theorized. Conflicting recommendations, duplicated tool calls, oscillating mitigation steps, and inconsistent constraint interpretations can turn modularity into fragility. A research need is auditable arbitration: mechanisms that resolve conflicts between agents based on explicit policies and evidence sufficiency criteria, with the resolution recorded in a trace that supports later inspection. This requires shared trace schemas, explicit responsibility boundaries, and evaluation metrics that quantify coordination failure rather than ignoring it as implementation noise [176,177].

11.4 Trace Schema Standardization, Interoperability, and Integrity Guarantees

Auditability depends on trace capture, but current practice is fragmented: systems log different artifacts, redact inconsistently, and rarely provide stable identifiers that support replay or third-party auditing. A core gap is the lack of interoperable trace schemas that connect evidence references, checks performed, tool interactions, and outcomes in a way that is query-able across tools and organizations [178]. Integrity is equally important. Trace records are attractive targets for tampering and audit evasion, and they can be corrupted accidentally through ingestion failures or retention policies. Research is needed on integrity-preserving trace pipelines, access control models for trace data, and principled redaction strategies that maintain audit usefulness while protecting sensitive data [143].

11.5 Verification Suites That Remain Valid under Drift and Evolving Systems

Verification mechanisms tend to degrade as environments evolve. Checks that once captured real preconditions can become shallow proxies, and dry-runs can drift away from real side effects as toolchains change [179]. The gap here is not simply that more checks are required, but maintainable verification suites with explicit versioning, coverage tracking, and drift monitoring [180]. A related need is methods that automatically detect when the check suite has become misaligned with the operational environment, using trace signals such as abnormal pass-rate shifts, increases in post-action regressions, or changes in evidence availability and structure [179].

11.6 Adversarial Robustness for Evidence Stores and Audit Pathways

Security research has identified many attack paths for tool-using agents, but the operational assurance literature still underweights attacks that target the evidence base and the audit pathway rather than the model directly. Prompt injection through retrieved artifacts, poisoning of runbooks and knowledge stores, log injection, and credential abuse all undermine assurance if provenance constraints, gating, and trace integrity are weak. The research gap is integrated defense: approaches that jointly secure evidence ingestion, retrieval scoping, tool privileges, and audit logs, while measuring how these defenses affect decision quality

and operational reliability. This includes explicit threat modeling for audit evasion and trace tampering, not merely for unsafe tool calls [181,182].

12 Conclusion

This review argued that operational autonomy for LLM-based agents is only defensible when it is auditable. The relevant standard is not whether an agent can produce plausible plans or complete tasks in a demonstration setting, but whether its decisions can be justified with time-relevant evidence, constrained by enforceable checks and gating, and reconstructed from an operational record that supports accountability. Framed this way, autonomy becomes something that must be operationally specified, governance-bounded and empirically evaluated rather than merely asserted through plausible narrative.

A central contribution of the paper is the separation of three planes, namely evidence, decision traces, and outcomes, and the assurance stack that operates across them. Evidence grounding is not equivalent to retrieval; it depends on provenance rules, freshness constraints, and sufficiency criteria that reflect partial observability. Verification is not a model introspection exercise; it is a set of logged, repeatable checks that can fail and must be treated as first-class evidence. Action gating is the boundary that makes risk-bounded autonomy possible, provided it is enforced at the tool and privilege level and aligned to the decision surface. Auditability and accountability controls make these mechanisms inspectable and contestable, enabling post-incident learning and preventing autonomy claims from resting on unverifiable explanations.

The synthesis also highlighted a practical implication that many studies still underreport: tool permissions, partial execution, and trace integrity are not secondary implementation details. They determine whether a system's behavior can be audited and whether evaluations can be interpreted across settings. Where these properties are unspecified, the most defensible interpretation is that the contribution is limited to decision support rather than operational autonomy, regardless of how an interface is presented.

Finally, the paper's research agenda emphasized that progress depends on shared evaluation infrastructure and on governance-aware trace practices. Comparable replay protocols, action-level benchmarks, maintainable verification suites under drift, and integrity-preserving trace schemas are prerequisites for moving from compelling demonstrations to reliable deployment. Without these foundations, autonomy claims will remain difficult to validate, easy to overstate, and costly to debug when failures occur.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: conceptualization, Leonidas Theodorakopoulos; methodology, Leonidas Theodorakopoulos; investigation, Leonidas Theodorakopoulos and Alexandra Theodoropoulou; writing—original draft preparation, Alexandra Theodoropoulou; writing—review and editing, Leonidas Theodorakopoulos and Alexandra Theodoropoulou; supervision, Leonidas Theodorakopoulos. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

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

References

1. Long S, Tan J, Mao B, Tang F, Li Y, Zhao M, et al. A survey on intelligent network operations and performance optimization based on large language models. IEEE Commun Surv Tutor. 2025;27(6):3915–49. doi:10.1109/comst.2025.3526606.
2. Chang Y, Wang X, Wang J, Wu Y, Yang L, Zhu K, et al. A survey on evaluation of large language models. ACM Trans Intell Syst Technol. 2024;15(3):1–45. doi:10.1145/3641289.
3. Mohammadi M, Li Y, Lo J, Yip W. Evaluation and benchmarking of LLM agents: a survey. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2. Toronto, ON, Canada. New York, NY, USA: ACM; 2025. p. 6129–39. doi:10.1145/3711896.3736570.
4. Wang L, Ma C, Feng X, Zhang Z, Yang H, Zhang J, et al. A survey on large language model based autonomous agents. Front Comput Sci. 2024;18(6):186345. doi:10.1007/s11704-024-40231-1.
5. Garg V. Designing the mind: how agentic frameworks are shaping the future of AI behavior. J Comput Sci Technol Stud. 2025;7(5):182–93. doi:10.32996/jcsts.2025.7.5.24.
6. Ananny M, Crawford K. Seeing without knowing: limitations of the transparency ideal and its application to algorithmic accountability. New Medium Soc. 2018;20(3):973–89. doi:10.1177/1461444816676645.
7. Shneiderman B. Bridging the gap between ethics and practice: guidelines for reliable, safe, and trustworthy human-centered AI systems. ACM Trans Interact Intell Syst. 2020;10(4):1–31. doi:10.1145/3419764.
8. Raji ID, Smart A, White RN, Mitchell M, Gebru T, Hutchinson B, et al. Closing the AI accountability gap: defining an end-to-end framework for internal algorithmic auditing. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. New York, NY, USA: ACM; 2020. p. 33–44. doi:10.1145/3351095.3372873.
9. Ojewale V, Steed R, Vecchione B, Birhane A, Raji ID. Towards AI accountability infrastructure: gaps and opportunities in AI audit tooling. In: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems. New York, NY, USA: ACM; 2025. p. 1–29. doi:10.1145/3706598.3713301.
10. Vassilev A, Oprea A, Fordyce A, Andersen H. Adversarial machine learning: a taxonomy and terminology of attacks and mitigations. Gaithersburg, MD, USA: National Institute of Standards and Technology; 2024. Report No.: NIST AI 100-2e2023. doi:10.6028/NIST.AI.100-2e2023.
11. Janssen M, Brous P, Estevez E, Barbosa LS, Janowski T. Data governance: organizing data for trustworthy Artificial Intelligence. Gov Inf Q. 2020;37(3):101493. doi:10.1016/j.giq.2020.101493.
12. Husák M, Sadlek L, Špaček S, Laštovička M, Javorník M, Komárková J. CRUSOE: a toolset for cyber situational awareness and decision support in incident handling. Comput Secur. 2022;115(5):102609. doi:10.1016/j.cose.2022.102609.
13. Amershi S, Weld D, Vorvoreanu M, Fourney A, Nushi B, Collisson P, et al. Guidelines for human-AI interaction. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. New York, NY, USA: ACM; 2019. p. 1–13. doi:10.1145/3290605.3300233.
14. Zhang Y, Dong C, Guo W, Dai J, Zhao Z. Systems theoretic accident model and process (STAMP): a literature review. Saf Sci. 2022;152(4):105596. doi:10.1016/j.ssci.2021.105596.
15. Lyell D, Coiera E. Automation bias and verification complexity: a systematic review. J Am Med Inform Assoc. 2017;24(2):423–31. doi:10.1093/jamia/ocw105.
16. Lauri M, Hsu D, Pajarinen J. Partially observable Markov decision processes in robotics: a survey. IEEE Trans Robot. 2023;39(1):21–40. doi:10.1109/TRO.2022.3200138.
17. Sterner P, Goretzko D, Pargent F. Everything has its price: foundations of cost-sensitive machine learning and its application in psychology. Psychol Methods. 2025;30(1):112–27. doi:10.1037/met0000586.
18. Li B, Peng X, Xiang Q, Wang H, Xie T, Sun J, et al. Enjoy your observability: an industrial survey of microservice tracing and analysis. Empir Softw Eng. 2021;27(1):25. doi:10.1007/s10664-021-10063-9.
19. Schelter S, Guha S, Grafberger S. Automated provenance-based screening of ML data preparation pipelines. Datenbank Spektrum. 2024;24(3):187–96. doi:10.1007/s13222-024-00483-4.
20. Aghili R, Li H, Khomh F. Protecting privacy in software logs: what should be anonymized? Proc ACM Softw Eng. 2025;2:1317–38. doi:10.1145/3715779.

21. Soriano-Salvador E, Guardiola-Múzquiz G. SealFS: storage-based tamper-evident logging. Comput Secur. 2021;108(7):102325. doi:10.1016/j.cose.2021.102325.
22. Dong Y, Mu R, Zhang Y, Sun S, Zhang T, Wu C, et al. Safeguarding large language models: a survey. Artif Intell Rev. 2025;58(12):382. doi:10.1007/s10462-025-11389-2.
23. Weidinger L, Uesato J, Rauh M, Griffin C, Huang PS, Mellor J, et al. Taxonomy of risks posed by language models. In: 2022 ACM Conference on Fairness Accountability and Transparency; 2022 Jun 21–24; Seoul, Republic of Korea. p. 214–29. doi:10.1145/3531146.3533088.
24. Yao Y, Duan J, Xu K, Cai Y, Sun Z, Zhang Y. A survey on large language model (LLM) security and privacy: the good, the bad, and the ugly. High Confid Comput. 2024;4(2):100211. doi:10.1016/j.hcc.2024.100211.
25. De la Cruz Cabello M, Sales TP, Machado MR. AIOps for log anomaly detection in the era of LLMs: a systematic literature review. Intell Syst Appl. 2025;28:200608. doi:10.1016/j.iswa.2025.200608.
26. Zhang L, Jia T, Jia M, Wu Y, Liu A, Yang Y, et al. A survey of AIOps in the era of large language models. ACM Comput Surv. 2026;58(2):1–35. doi:10.1145/3746635.
27. Ferrag MA, Tihanyi N, Hamouda D, Maglaras L, Lakas A, Debbah M. From prompt injections to protocol exploits: threats in LLM-powered AI agents workflows. ICT Express. 2026;12(2):353–83. doi:10.1016/j.icte.2025.12.001.
28. Díaz-Rodríguez N, Del Ser J, Coeckelbergh M, López de Prado M, Herrera-Viedma E, Herrera F. Connecting the dots in trustworthy Artificial Intelligence: from AI principles, ethics, and key requirements to responsible AI systems and regulation. Inf Fusion. 2023;99(2):101896. doi:10.1016/j.inffus.2023.101896.
29. Novelli C, Casolari F, Rotolo A, Taddeo M, Floridi L. AI risk assessment: a scenario-based, proportional methodology for the AI act. Digit Soc. 2024;3(1):13. doi:10.1007/s44206-024-00095-1.
30. Raji ID, Kumar IE, Horowitz A, Selbst A. The fallacy of AI functionality. In: 2022 ACM Conference on Fairness Accountability and Transparency. Seoul, Republic of Korea: ACM; 2022. p. 959–72. doi:10.1145/3531146.3533158.
31. Zhang S, Xia S, Fan W, Shi B, Xiong X, Zhong Z, et al. Failure diagnosis in microservice systems: a comprehensive survey and analysis. ACM Trans Softw Eng Methodol. 2026;35(1):1–55. doi:10.1145/3715005.
32. Sadlek L, Yamin MM, Čeleda P, Katt B. Severity-based triage of cybersecurity incidents using kill chain attack graphs. J Inf Secur Appl. 2025;89(2):103956. doi:10.1016/j.jisa.2024.103956.
33. Ahmed T, Ghosh S, Bansal C, Zimmermann T, Zhang X, Rajmohan S. Recommending root-cause and mitigation steps for cloud incidents using large language models. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE); 2023 May 14–20; Melbourne, Australia. p. 1737–49. doi:10.1109/ICSE48619.2023.00149.
34. Butt AS, Fitch P. A provenance model for control-flow driven scientific workflows. Data Knowl Eng. 2021;131–132(1):101877. doi:10.1016/j.datak.2021.101877.
35. Mökander J, Axente M. Ethics-based auditing of automated decision-making systems: intervention points and policy implications. AI Soc. 2023;38(1):153–71. doi:10.1007/s00146-021-01286-x.
36. Recupito G, Pecorelli F, Catolino G, Lenarduzzi V, Taibi D, Di Nucci D, et al. Technical debt in AI-enabled systems: on the prevalence, severity, impact, and management strategies for code and architecture. J Syst Softw. 2024;216(3):112151. doi:10.1016/j.jss.2024.112151.
37. Varshney KR, Alemzadeh H. On the safety of machine learning: cyber-physical systems, decision sciences, and data products. Big Data. 2017;5(3):246–55. doi:10.1089/big.2016.0051.
38. Shahin M, Ali Babar M, Zahedi M, Zhu L. Beyond continuous delivery: an empirical investigation of continuous deployment challenges. In: 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM); 2017 Nov 9–10; Toronto, ON, Canada. p. 111–20. doi:10.1109/ESEM.2017.18.
39. Wu Y, Chai B, Li Y, Liu B, Li J, Yang Y, et al. An empirical study on change-induced incidents of online service systems. In: 2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP); 2023 May 14–20; Melbourne, Australia. p. 234–45. doi:10.1109/ICSE-SEIP58684.2023.00027.
40. Stemn E, Bofinger C, Cliff D, Hassall ME. Failure to learn from safety incidents: status, challenges and opportunities. Saf Sci. 2018;101:313–25. doi:10.1016/j.ssci.2017.09.018.
41. Leo M, Tan F, Miao T, Anand G. From threat to trust: assessing security risks of agentic AI systems. Int J Inf Secur. 2026;25(1):23. doi:10.1007/s10207-025-01185-y.

42. Zhang Q, Fu L, Lian L, Go G, Wang Y, Zhou C, et al. Evaluating privilege usage of agents on real-world tools. In: Proceedings of the 34th ACM Symposium on User Interface Software and Technology; 2026 Nov 2–5; Detroit, MI, USA. p. 414:1–22. doi:10.1145/3706598.3713218.
43. Reale C, Salwei ME, Militello LG, Weinger MB, Burden A, Sushereba C, et al. Decision-making during high-risk events: a systematic literature review. J Cogn Eng Decis Mak. 2023;17(2):188–212. doi:10.1177/15553434221147415.
44. Wang W, Chen J, Yang L, Zhang H, Wang Z. Understanding and predicting incident mitigation time. Inf Softw Technol. 2023;155(4):107119. doi:10.1016/j.infsof.2022.107119.
45. Kroll JA. Outlining traceability: a principle for operationalizing accountability in computing systems. In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. Virtual. p. 758–71. doi:10.1145/3442188.3445937.
46. Lazcoz G, de Hert P. Humans in the GDPR and AIA governance of automated and algorithmic systems. Essential pre-requisites against abdicating responsibilities. Comput Law Secur Rev. 2023;50:105833. doi:10.1016/j.clsr.2023.105833.
47. He F, Zhu T, Ye D, Liu B, Zhou W, Yu PS. The emerged security and privacy of LLM agent: a survey with case studies. ACM Comput Surv. 2026;58(6):1–36. doi:10.1145/3773080.
48. Hobbs KL, Mote ML, Abate MCL, Coogan SD, Feron EM. Runtime assurance for safety-critical systems: an introduction to safety filtering approaches for complex control systems. IEEE Control Syst Mag. 2023;43(2):28–65. doi:10.1109/MCS.2023.3234380.
49. Werder K, Ramesh B, Zhang RS. Establishing data provenance for responsible artificial intelligence systems. ACM Trans Manage Inf Syst. 2022;13(2):1–23. doi:10.1145/3503488.
50. Putz B, Menges F, Pernul G. A secure and auditable logging infrastructure based on a permissioned blockchain. Comput Secur. 2019;87(7–8):101602. doi:10.1016/j.cose.2019.101602.
51. Schröder T, Schulz M. Monitoring machine learning models: a categorization of challenges and methods. Data Sci Manag. 2022;5(3):105–16. doi:10.1016/j.dsm.2022.07.004.
52. Goel K, Martin N, ter Hofstede A. Demystifying data governance for process mining: insights from a Delphi study. Inf Manag. 2024;61(5):103973. doi:10.1016/j.im.2024.103973.
53. Kale A, Nguyen T, Harris FC Jr, Li C, Zhang J, Ma X. Provenance documentation to enable explainable and trustworthy AI: a literature review. Data Intell. 2023;5(1):139–62. doi:10.1162/dint_a_00119.
54. Zhou X, Peng X, Xie T, Sun J, Ji C, Li W, et al. Fault analysis and debugging of microservice systems: industrial survey, benchmark system, and empirical study. IIEEE Trans Software Eng. 2021;47(2):243–60. doi:10.1109/tse.2018.2887384.
55. Bento A, Correia J, Filipe R, Araujo F, Cardoso J. Automated analysis of distributed tracing: challenges and research directions. J Grid Comput. 2021;19(1):9. doi:10.1007/s10723-021-09551-5.
56. Hammad Y, Al-Said Ahmad A, Andras P. An empirical study on the performance overhead of code instrumentation in containerised microservices. J Syst Softw. 2025;230(9):112573. doi:10.1016/j.jss.2025.112573.
57. Tang Y, Spektor A, Khatchadourian R, Bagherzadeh M. Automated evolution of feature logging statement levels using Git histories and degree of interest. Sci Comput Program. 2022;214:102724. doi:10.1016/j.scico.2021.102724.
58. Drosos GP, Sotiropoulos T, Alexopoulos G, Mitropoulos D, Su Z. When your infrastructure is a buggy program: understanding faults in infrastructure as code ecosystems. Proc ACM Program Lang. 2024;8(OOPSLA2):2490–520. doi:10.1145/3689799.
59. Shaked A, Cherdantseva Y, Burnap P, Maynard P. Operations-informed incident response playbooks. Comput Secur. 2023;134:103454. doi:10.1016/j.cose.2023.103454.
60. Zhang Z, Wang J, Li B, Zhang L, Liu Y, Hung P. ADmM: anomaly detection for microservice systems with incomplete metrics. ACM Trans Web. 2026;20(2):1–35. doi:10.1145/3801729.
61. Singh J, Cobbe J, Norval C. Decision provenance: harnessing data flow for accountable systems. IEEE Access. 2019;7:6562–74. doi:10.1109/ACCESS.2018.2887201.
62. Schlegel M, Sattler KU. Capturing end-to-end provenance for machine learning pipelines. Inf Syst. 2025;132(13):102495. doi:10.1016/j.is.2024.102495.

63. Varanda A, Santos L, de C Costa RL, Oliveira A, Rabadão C. Log pseudonymization: privacy maintenance in practice. J Inf Secur Appl. 2021;63(2):103021. doi:10.1016/j.jisa.2021.103021.
64. Guardiola-Múzquiz G, Soriano-Salvador E. SealFSv2: combining storage-based and ratcheting for tamper-evident logging. Int J Inf Secur. 2023;22(2):447–66. doi:10.1007/s10207-022-00643-1.
65. Nõu A, Talluri S, Iosup A, Bonetta D. Investigating performance overhead of distributed tracing in microservices and serverless systems. In: Companion of the 16th ACM/SPEC International Conference on Performance Engineering. Toronto, ON, Canada: ACM; 2025. p. 162–6. doi:10.1145/3680256.3721316.
66. Xie Y, Feng D, Liao X, Qin L. Efficient monitoring and forensic analysis via accurate network-attached provenance collection with minimal storage overhead. Digit Investig. 2018;26(1):19–28. doi:10.1016/j.diin.2018.05.001.
67. Rupprecht L, Davis JC, Arnold C, Gur Y, Bhagwat D. Improving reproducibility of data science pipelines through transparent provenance capture. Proc VLDB Endow. 2020;13(12):3354–68. doi:10.14778/3415478.3415556.
68. Isaac Abiodun O, Alawida M, Esther Omolara A, Alabdulatif A. Data provenance for cloud forensic investigations, security, challenges, solutions and future perspectives: a survey. J King Saud Univ Comput Inf Sci. 2022;34(10):10217–45. doi:10.1016/j.jksuci.2022.10.018.
69. Kumar A, Nadeem M, Shameem M. A systematic literature review for investigating DevOps metrics to implement in software development organizations. J Software Evolu Process. 2025;37(1):e2733. doi:10.1002/smr.2733.
70. Amaro R, Pereira R, Mira da Silva M. DevOps metrics and KPIs: a multivocal literature review. ACM Comput Surv. 2024;56(9):1–41. doi:10.1145/3652508.
71. van der Kleij R, Schraagen JM, Cadet B, Young H. Developing decision support for cybersecurity threat and incident managers. Comput Secur. 2022;113(8):102535. doi:10.1016/j.cose.2021.102535.
72. Xin R, Chen P, Zhao Z. CausalRCA: causal inference based precise fine-grained root cause localization for microservice applications. J Syst Softw. 2023;203(3):111724. doi:10.1016/j.jss.2023.111724.
73. Palma A, Acitelli G, Marrella A, Bonomi S, Angelini M. A compliance assessment system for incident management process. Comput Secur. 2024;146(10):104070. doi:10.1016/j.cose.2024.104070.
74. Huang D, Li J, Cai H, Li X, Xu Y, Tang W, et al. A vision for access control in LLM agent systems. In: Engineering of Complex Computer Systems. Cham, Switzerland: Springer Nature; 2025. p. 467–72. doi:10.1007/978-3-032-00828-2_25.
75. Doshi A, Hong Y, Xu C, Kang E, Kapravelos A, Kästner C. Towards verifiably safe tool use for LLM agents. In: Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER '26); 2026 Apr 14–18; Rio de Janeiro, Brazil. p. 1–5. doi:10.48550/arXiv.2601.08012.
76. Hu H, Chen P, Zhao Y, Chen Y. AgentSentinel: an end-to-end and real-time security defense framework for computer-use agents. In: Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security; 2025 Oct 13–17; Taipei, Taiwan. p. 3535–49. doi:10.1145/3719027.3765064.
77. Liu Y, Lo SK, Lu Q, Zhu L, Zhao D, Xu X, et al. Agent design pattern catalogue: a collection of architectural patterns for foundation model based agents. J Syst Softw. 2025;220(5):112278. doi:10.1016/j.jss.2024.112278.
78. Zhai W, Liao J, Chen Z, Su B, Zhao X. A survey of task planning with large language models. Intell Comput. 2025;4(1):124. doi:10.34133/icomputing.0124.
79. Zouari J. Toward agentic IAM: a probabilistic authorization framework for least privilege AI workflows. In: Proceedings of the IEEE/ACM 12th International Conference on Big Data Computing, Applications and Technologies; 2025 Dec 1–4; Nantes, France. p. 1–6. doi:10.1145/3773276.3776564.
80. Ntousakis G, Stephen JJ, Le MV, Chukkapalli SSL, Taylor T, Molloy IM, et al. Securing MCP-based agent workflows. In: Proceedings of the 4th Workshop on Practical Adoption Challenges of ML for Systems; 2025 Oct 13–16; Seoul, Republic of Korea. p. 50–5. doi:10.1145/3766882.3767177.
81. Klesel M, Wittmann HF. Retrieval-augmented generation (RAG). Bus Inf Syst Eng. 2025;67(4):551–61. doi:10.1007/s12599-025-00945-3.
82. Ji Z, Lee N, Frieske R, Yu T, Su D, Xu Y, et al. Survey of hallucination in natural language generation. ACM Comput Surv. 2023;55(12):1–38. doi:10.1145/3571730.
83. Li X, Wang S, Zeng S, Wu Y, Yang Y. A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. Vicinagearth. 2024;1(1):9. doi:10.1007/s44336-024-00009-2.

84. Sapkota R, Roumeliotis KI, Karkee M. AI Agents vs. Agentic AI: a conceptual taxonomy, applications and challenges. Inf Fusion. 2026;126(1–2):103599. doi:10.1016/j.inffus.2025.103599.
85. Neto AVS, Camargo JB, Almeida JR, Cugnasca PS. Safety assurance of artificial intelligence-based systems: a systematic literature review on the state of the art and guidelines for future work. IEEE Access. 2022;10(109):130733–70. doi:10.1109/ACCESS.2022.3229233.
86. Meroño-Peñuela A, Simperl E, Kurteva A, Reklos I. KG.GOV: knowledge graphs as the backbone of data governance in AI. J Web Semant. 2025;85(3):100847. doi:10.1016/j.websem.2024.100847.
87. Jacques-Silva G, Kalyvianaki E, Cohn-Gordon K, Meguid A, Nguyen H, Ben-David D, et al. Unified lineage system: tracking data provenance at scale. In: Companion of the 2025 International Conference on Management of Data; 2025 Jun 22–27; Berlin, Germany. p. 457–70. doi:10.1145/3722212.3724458.
88. Chen Q, Wang T, Legunsen O, Li S, Xu T. Understanding and discovering software configuration dependencies in cloud and datacenter systems. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering; 2020 Nov 8–13; Virtual Event. p. 362–74. doi:10.1145/3368089.3409727.
89. Henriques J, Caldeira F, Cruz T, Simões P. An automated closed-loop framework to enforce security policies from anomaly detection. Comput Secur. 2022;123(3):102949. doi:10.1016/j.cose.2022.102949.
90. Rose S, Borchert O, Mitchell S, Connelly S. Zero trust architecture. NIST Spec Publ. 2020;800(207):1–52. doi:10.6028/NIST.SP.800-207.
91. Deng Z, Guo Y, Han C, Ma W, Xiong J, Wen S, et al. AI agents under threat: a survey of key security challenges and future pathways. ACM Comput Surv. 2025;57(7):1–36. doi:10.1145/3716628.
92. Ali A, Ahmed M, Khan A. Audit logs management and security—a survey. Kuwait J Sci. 2021;48(3):1–18. doi:10.48129/kjs.v48i3.10624.
93. Wei R, Foster S, Mei H, Yan F, Yang R, Habli I, et al. ACCESS: assurance case centric engineering of safety–critical systems. J Syst Softw. 2024;213(1):112034. doi:10.1016/j.jss.2024.112034.
94. Ahlbrecht A, Sprockhoff J, Durak U. A system-theoretic assurance framework for safety-driven systems engineering. Softw Syst Model. 2025;24(1):253–70. doi:10.1007/s10270-024-01209-6.
95. Pasquier T, Singh J, Powles J, Eyers D, Seltzer M, Bacon J. Data provenance to audit compliance with privacy policy in the Internet of Things. Pers Ubiquitous Comput. 2018;22(2):333–44. doi:10.1007/s00779-017-1067-4.
96. Wallat J, Heuss M, de Rijke M, Anand A. Correctness is not faithfulness in retrieval augmented generation attributions. In: Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR); 2025 Jul 18; Padua, Italy. p. 22–32. doi:10.1145/3731120.3744592.
97. Amalfitano D, De Luca M, Rita Fasolino A, Tramontana P. Statistical-based metric threshold setting method for software fault prediction in firmware projects: an industrial experience. J Syst Softw. 2026;236(11):112818. doi:10.1016/j.jss.2026.112818.
98. Bates S, Angelopoulos A, Lei L, Malik J, Jordan M. Distribution-free, risk-controlling prediction sets. J ACM. 2021;68(6):1–34. doi:10.1145/3478535.
99. Sokolowski D, Spielmann D, Salvaneschi G. Automated infrastructure as code program testing. IEEE Trans Softw Eng. 2024;50(6):1585–99. doi:10.1109/TSE.2024.3393070.
100. Kim Y, Dán G, Zhu Q. Human-in-the-loop cyber intrusion detection using active learning. IEEE Trans Inf Forensics Secur. 2024;19:8658–72. doi:10.1109/TIFS.2024.3434647.
101. Barandas M, Folgado D, Santos R, Simão R, Gamboa H. Uncertainty-based rejection in machine learning: implications for model development and interpretability. Electronics. 2022;11(3):396. doi:10.3390/electronics11030396.
102. Lee S, Huh JH, Woo H. Security system design and verification for zero trust architecture. Electronics. 2025;14(4):643. doi:10.3390/electronics14040643.
103. Tsai L, Bagdasarian E. Contextual agent security: a policy for every purpose. In: Proceedings of the Workshop on Hot Topics in Operating Systems; 2025 May 14–16; Banff, AB, Canada. p. 8–17. doi:10.1145/3713082.3730378.
104. Mora-Cantallops M, Sánchez-Alonso S, García-Barriocanal E, Sicilia MA. Traceability for trustworthy AI: a review of models and tools. Big Data Cogn Comput. 2021;5(2):20. doi:10.3390/bdcc5020020.

105. Paleyes A, Urma RG, Lawrence ND. Challenges in deploying machine learning: a survey of case studies. ACM Comput Surv. 2023;55(6):1–29. doi:10.1145/3533378.
106. Kapoor S, Narayanan A. Leakage and the reproducibility crisis in machine-learning-based science. Patterns. 2023;4(9):100804. doi:10.1016/j.patter.2023.100804.
107. Ji Y, Sun A, Zhang J, Li C. A critical study on data leakage in recommender system offline evaluation. ACM Trans Inf Syst. 2023;41(3):1–27. doi:10.1145/3569930.
108. Theodorakopoulos L, Karras A, Theodoropoulou A, Kampiotis G. Benchmarking big data systems: performance and decision-making implications in emerging technologies. Technologies. 2024;12(11):217. doi:10.3390/technologies12110217.
109. Palacios Chavarro S, Nespoli P, Díaz-López D, Niño Roa Y. On the way to automatic exploitation of vulnerabilities and validation of systems security through security chaos engineering. Big Data Cogn Comput. 2023;7(1):1. doi:10.3390/bdcc7010001.
110. Cief M, Kveton B, Kompan M. Cross-validated off-policy evaluation. Proc AAAI Conf Artif Intell. 2025;39(15):16073–81. doi:10.1609/aaai.v39i15.33765.
111. Quin F, Weyns D, Galster M, Silva CC. A/B testing: a systematic literature review. J Syst Softw. 2024;211(3):112011. doi:10.1016/j.jss.2024.112011.
112. Sheng J, Liu H, Wang B. Research on the optimization of A/B testing system based on dynamic strategy distribution. Processes. 2023;11(3):912. doi:10.3390/pr11030912.
113. Wang H, Huang W, Wang Y, Xi Y, Lu J, Zhang H, et al. Rethinking stateful tool use in multi-turn dialogues: benchmarks and challenges. In: Findings of the Association for Computational Linguistics: ACL 2025; 2025 Jul 27–Aug 1; Vienna, Austria. p. 5433–53. doi:10.18653/v1/2025.findings-acl.284.
114. Balaji S, Mishra P, Sachdeva A, Agrawal S. Beyond IVR: benchmarking customer support LLM agents for business-adherence. In: Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track). Stroudsburg, PA, USA: ACL; 2026. p. 193–208. doi:10.18653/v1/2026.eacl-industry.15.
115. Kourani H, Berti A, Schuster D, van der Aalst WMP. Evaluating large language models on business process modeling: framework, benchmark, and self-improvement analysis. Softw Syst Model. 2025;1–36. doi:10.1007/s10270-025-01318-w.
116. Patcas R, Motogna S. An evaluation study of large language models for addressing code quality issues. Empir Softw Eng. 2026;31(5):118. doi:10.1007/s10664-026-10858-8.
117. Pahune S, Akhtar Z. Transitioning from MLOps to LLMOps: navigating the unique challenges of large language models. Information. 2025;16(2):87. doi:10.3390/info16020087.
118. Hutchinson B, Rostamzadeh N, Greer C, Heller K, Prabhakaran V. Evaluation gaps in machine learning practice. In: 2022 ACM Conference on Fairness Accountability and Transparency. Seoul, Republic of Korea: ACM; 2022. p. 1859–76. doi:10.1145/3531146.3533233.
119. Nastoska A, Jancheska B, Rizinski M, Trajanov D. Evaluating trustworthiness in AI: risks, metrics, and applications across industries. Electronics. 2025;14(13):2717. doi:10.3390/electronics14132717.
120. Xu D, Gondal I, Yi X, Susnjak T, Watters P, McIntosh TR. The erosion of cybersecurity zero-trust principles through generative AI: a survey on the challenges and future directions. J Cybersecur Priv. 2025;5(4):87. doi:10.3390/jcp5040087.
121. Morillo Reina JD, Mateo Sanguino TJ. Decentralized and secure blockchain solution for tamper-proof logging events. Future Internet. 2025;17(3):108. doi:10.3390/fi17030108.
122. Gulyamov S, Gulyamov S, Rodionov A, Khursanov R, Mekhmonov K, Babaev D, et al. Prompt injection attacks in large language models and AI agent systems: a comprehensive review of vulnerabilities, attack vectors, and defense mechanisms. Information. 2026;17(1):54. doi:10.3390/info17010054.
123. Shisher MKC, Sun Y. How does data freshness affect real-time supervised learning? In: Proceedings of the Twenty-Third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing; 2022 Oct 17–20; Seoul, Republic of Korea. p. 31–40. doi:10.1145/3492866.3549711.

124. Gawlikowski J, Tassi CRN, Ali M, Lee J, Humt M, Feng J, et al. A survey of uncertainty in deep neural networks. Artif Intell Rev. 2023;56(S1):1513–89. doi:10.1007/s10462-023-10562-9.
125. Hendrickx K, Perini L, Van der Plas D, Meert W, Davis J. Machine learning with a reject option: a survey. Mach Learn. 2024;113(5):3073–110. doi:10.1007/s10994-024-06534-x.
126. Shekhtman L, Waisbard E. EngraveChain: a blockchain-based tamper-proof distributed log system. Fut Internet. 2021;13(6):143. doi:10.3390/fi13060143.
127. Zhang B, Tan Y, Shen Y, Salem A, Backes M, Zannettou S, et al. Breaking agents: compromising autonomous LLM agents through malfunction amplification. In: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing; 2025 Nov 4–9; Suzhou, China. p. 34964–76.
128. Suárez-Cetrulo AL, Quintana D, Cervantes A. A survey on machine learning for recurring concept drifting data streams. Expert Syst Appl. 2023;213(1):118934. doi:10.1016/j.eswa.2022.118934.
129. Khritankov A. Positive feedback loops lead to concept drift in machine learning systems. Appl Intell. 2023;53(19):22648–66. doi:10.1007/s10489-023-04615-3.
130. Hardt M, Mendler-Dünner C. Performative prediction: past and future. Statist Sci. 2025;40(3):419–36. doi:10.1214/25-sts986.
131. Liao Z, Chen K, Lin Y, Li K, Liu Y, Chen H, et al. Attack and defense techniques in large language models: a survey and new perspectives. Neural Netw. 2026;196(3):108388. doi:10.1016/j.neunet.2025.108388.
132. Karras A, Theodorakopoulos L, Karras C, Theodoropoulou A, Kalliampakou I, Kalogeratos G. LLMs for cybersecurity in the big data era: a comprehensive review of applications, challenges, and future directions. Information. 2025;16(11):957. doi:10.3390/info16110957.
133. Das BC, Amini MH, Wu Y. Security and privacy challenges of large language models: a survey. ACM Comput Surv. 2025;57(6):1–39. doi:10.1145/3712001.
134. Yerlikaya FA, Bahtiyar Ş. Data poisoning attacks against machine learning algorithms. Expert Syst Appl. 2022;208:118101. doi:10.1016/j.eswa.2022.118101.
135. Zhang B, Xin H, Fang M, Liu Z. Traceback of poisoning attacks to retrieval-augmented generation. In: Proceedings of the ACM on Web Conference 2025; 2025 Apr 28; Sydney, NSW, Australia. p. 2085–97. doi:10.1145/3696410.3714756.
136. Zhang W, Zhang J. Hallucination mitigation for retrieval-augmented large language models: a review. Mathematics. 2025;13(5):856. doi:10.3390/math13050856.
137. Landauer M, Onder S, Skopik F, Wurzenberger M. Deep learning for anomaly detection in log data: a survey. Mach Learn Appl. 2023;12(3):100470. doi:10.1016/j.mlwa.2023.100470.
138. Tan L, Su W, Zhang W, Lv J, Zhang Z, Miao J, et al. In-band network telemetry: a survey. Comput Netw. 2021;186(3):107763. doi:10.1016/j.comnet.2020.107763.
139. Pei J, Hu Y, Tian L, Pei X, Wang Z. Dynamic anomaly detection using in-band network telemetry and GCN for cloud–edge collaborative networks. Comput Secur. 2025;154(2):104422. doi:10.1016/j.cose.2025.104422.
140. Regueiro C, Seco I, Gutiérrez-Agüero I, Urquizu B, Mansell J. A blockchain-based audit trail mechanism: design and implementation. Algorithms. 2021;14(12):341. doi:10.3390/a14120341.
141. Wang X, Yang X, Liang X, Zhang X, Zhang W, Gong X. Combating alert fatigue with AlertPro: context-aware alert prioritization using reinforcement learning for multi-step attack detection. Comput Secur. 2024;137(1):103583. doi:10.1016/j.cose.2023.103583.
142. Ultra JD, Pancho-Festin S. A simple model of separation of duty for access control models. Comput Secur. 2017;68:69–80. doi:10.1016/j.cose.2017.03.012.
143. Yagemann C, Noureddine MA, Hassan WU, Chung S, Bates A, Lee W. Validating the integrity of audit logs against execution repartitioning attacks. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security; 2021 Nov 15–19; Virtual Event. p. 3337–51. doi:10.1145/3460120.3484551.
144. Zafar F, Khan A, Suhail S, Ahmed I, Hameed K, Khan HM, et al. Trustworthy data: a survey, taxonomy and future trends of secure provenance schemes. J Netw Comput Appl. 2017;94(2):50–68. doi:10.1016/j.jnca.2017.06.003.
145. Elouataoui W, El Alaoui I, El Mendili S, Gahi Y. An advanced big data quality framework based on weighted metrics. Big Data Cogn Comput. 2022;6(4):153. doi:10.3390/bdcc6040153.

146. Golightly L, Modesti P, Garcia R, Chang V. Securing distributed systems: a survey on access control techniques for cloud, blockchain, IoT and SDN. Cyber Secur Appl. 2023;1(7):100015. doi:10.1016/j.csa.2023.100015.
147. Theodorakopoulos L, Theodoropoulou A, Halkiopoulos C. Enhancing decentralized decision-making with big data and blockchain technology: a comprehensive review. Appl Sci. 2024;14(16):7007. doi:10.3390/app14167007.
148. Waltersdorfer L, Sabou M. Leveraging knowledge graphs for AI system auditing and transparency. J Web Semant. 2025;84(2):100849. doi:10.1016/j.websem.2024.100849.
149. Blass EO, Noubir G. Forward security with crash recovery for secure logs. ACM Trans Priv Secur. 2024;27(1):1–28. doi:10.1145/3631524.
150. Ghosh S, Shetty M, Bansal C, Nath S. How to fight production incidents?: an empirical study on a large-scale cloud service. In: Proceedings of the 13th Symposium on Cloud Computing; 2022 Nov 7–11; San Francisco, CA, USA. p. 126–41. doi:10.1145/3542929.3563482.
151. Arfan Uddin M, Weerasinghe S, Gajewski D, Akbarsharifi M, Akbarsharifi R, Stoner C, et al. Microservice logs analysis employing AI: a systematic literature review. J Syst Softw. 2026;236(2):112786. doi:10.1016/j.jss.2026.112786.
152. Port D, Taber B, Emkani P. Investigating effectiveness and compliance to DevOps policies and practices for managing productivity and quality variability. J Syst Softw. 2024;213(7):112030. doi:10.1016/j.jss.2024.112030.
153. Theodorakopoulos L, Theodoropoulou A, Stamatiou Y. A state-of-the-art review in big data management engineering: real-life case studies, challenges, and future research directions. Eng. 2024;5(3):1266–97. doi:10.3390/eng5030068.
154. Derner E, Batistič K, Zahálka J, Babuška R. A security risk taxonomy for prompt-based interaction with large language models. IEEE Access. 2024;12(7):126176–87. doi:10.1109/ACCESS.2024.3450388.
155. Bridges RA, Rice AE, Oesch S, Nichols JA, Watson C, Spakes K, et al. Testing SOAR tools in use. Comput Secur. 2023;129(1):103201. doi:10.1016/j.cose.2023.103201.
156. Ahmad A, Saad M, Mohaisen A. Secure and transparent audit logs with BlockAudit. J Netw Comput Appl. 2019;145(2):102406. doi:10.1016/j.jnca.2019.102406.
157. Bernardo BMV, Mamede HS, Barroso JMP, dos Santos VMPD. Data governance & quality management—innovation and breakthroughs across different fields. J Innov Knowl. 2024;9(4):100598. doi:10.1016/j.jik.2024.100598.
158. Chen Y, Zhao Y, Li X, Zhang J, Long J, Zhou F. An open dataset of data lineage graphs for data governance research. Vis Inform. 2024;8(1):1–5. doi:10.1016/j.visinf.2024.01.001.
159. Pelosi D, Cacciagrano D, Piangerelli M. Explainability and interpretability in concept and data drift: a systematic literature review. Algorithms. 2025;18(7):443. doi:10.3390/a18070443.
160. El-Kady AH, Halim S, El-Halwagi MM, Khan F. Analysis of safety and security challenges and opportunities related to cyber-physical systems. Process Saf Environ Prot. 2023;173:384–413. doi:10.1016/j.psep.2023.03.012.
161. Furrer FJ. Safe and secure system architectures for cyber-physical systems. Inf Spektrum. 2023;46(2):96–103. doi:10.1007/s00287-023-01533-z.
162. Liu M, Zhang L, Xu W, Jiang S, Kong F. CPSim: simulation toolbox for security problems in cyber-physical systems. ACM Trans Des Autom Electron Syst. 2024;29(5):1–16. doi:10.1145/3674904.
163. Ledro C, Nosella A, Vinelli A, Dalla Pozza I, Souverain T. Artificial intelligence in customer relationship management: a systematic framework for a successful integration. J Bus Res. 2025;199:115531. doi:10.1016/j.jbusres.2025.115531.
164. Palma A, Angelini M. IMPAVID: enhancing incident management process compliance assessment with visual analytics. Comput Graph. 2025;130(5):104243. doi:10.1016/j.cag.2025.104243.
165. Sifakis J, Harel D. Trustworthy autonomous system development. ACM Trans Embed Comput Syst. 2023;22(3):1–24. doi:10.1145/3545178.
166. Servos D, Osborn SL. Current research and open problems in attribute-based access control. ACM Comput Surv. 2017;49(4):1–45. doi:10.1145/3007204.
167. Arnold M, Bellamy RKE, Hind M, Houde S, Mehta S, Mojsilović A, et al. FactSheets: increasing trust in AI services through supplier's declarations of conformity. IBM J Res Dev. 2019;63(4/5):6:1–613. doi:10.1147/jrd.2019.2942288.

168. Paterson C, Hawkins R, Picardi C, Jia Y, Calinescu R, Habli I. Safety assurance of machine learning for autonomous systems. Reliab Eng Syst Saf. 2025;264(1):111311. doi:10.1016/j.ress.2025.111311.
169. Zheng Y, Chen Y, Qian B, Shi X, Shu Y, Chen J. A review on edge large language models: design, execution, and applications. ACM Comput Surv. 2025;57(8):1–35. doi:10.1145/3719664.
170. Li X, Lu Z, Cai D, Ma X, Xu M. Large language models on mobile devices: measurements, analysis, and insights. In: Proceedings of the Workshop on Edge and Mobile Foundation Models; 2024 Jun 3–7; Tokyo, Japan. p. 1–6. doi:10.1145/3662006.3662059.
171. Piccialli F, Chiaro D, Qi P, Bellandi V, Damiani E. Federated and edge learning for large language models. Inf Fusion. 2025;117(1):102840. doi:10.1016/j.inffus.2024.102840.
172. Pan B, Stakhanova N, Ray S. Data provenance in security and privacy. ACM Comput Surv. 2023;55(14s):1–35. doi:10.1145/3593294.
173. Salah S, Maciá-Fernández G, Díaz-Verdejo JE. Fusing information from tickets and alerts to improve the incident resolution process. Inf Fusion. 2019;45(6):38–52. doi:10.1016/j.inffus.2018.01.011.
174. Graf-Vlachy L, Wagner S. Cleaning up confounding: accounting for endogeneity using instrumental variables and two-stage models. ACM Trans Softw Eng Methodol. 2024;33(8):1–31. doi:10.1145/3674730.
175. Hemming K, Haines TP, Chilton PJ, Girling AJ, Lilford RJ. The stepped wedge cluster randomised trial: rationale, design, analysis, and reporting. BMJ. 2015;350:h391. doi:10.1136/bmj.h391.
176. Baldoni M, Baroglio C, Micalizio R, Tedeschi S. Accountability in multi-agent organizations: from conceptual design to agent programming. Auton Agents Multi Agent Syst. 2022;37(1):7. doi:10.1007/s10458-022-09590-6.
177. Bandi A, Kongari B, Naguru R, Pasnoor S, Vilipala SV. The rise of agentic AI: a review of definitions, frameworks, architectures, applications, evaluation metrics, and challenges. Future Internet. 2025;17(9):404. doi:10.3390/fi17090404.
178. Janes A, Li X, Lenarduzzi V. Open tracing tools: overview and critical comparison. J Syst Softw. 2023;204(1):111793. doi:10.1016/j.jss.2023.111793.
179. Guan K, Legunsen O. Instrumentation-driven evolution-aware runtime verification. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE); 2025 Apr 26–May 6; Ottawa, ON, Canada. p. 103–15. doi:10.1109/ICSE55347.2025.00099.
180. Shimmi S, Rahimi M. Patterns of code-to-test co-evolution for automated test suite maintenance. In: 2022 IEEE Conference on Software Testing, Verification and Validation (ICST); 2022 Apr 4–14; Valencia, Spain. p. 116–27. doi:10.1109/ICST53961.2022.00023.
181. Yang R, Fu M, Tantithamthavorn C, Arora C, Vandenhurk L, Chua J. RAGVA: engineering retrieval augmented generation-based virtual assistants in practice. J Syst Softw. 2025;226(8):112436. doi:10.1016/j.jss.2025.112436.
182. Zhao T, Chen J, Ru Y, Zhu H, Hu N, Liu J, et al. Exploring knowledge poisoning attacks to retrieval-augmented generation. Inf Fusion. 2026;127(10):103900. doi:10.1016/j.inffus.2025.103900.