Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.081005

REVIEW

From Static to Streaming: A Systematic Review and Event-Sourced Framework for GraphRAG in AIOps

Ferenc Erdős[1,*], Vijayakumar Varadarajan[2,3,4], Viorel-Costin Banţa[5] and Stephen Afrifa[6,7]

[1]Department of Informatics, Faculty of Informatics and Electrical Engineering, Széchenyi István University, Győr, Hungary
[2]Swiss School of Business and Management Geneva, Geneva, Switzerland
[3]Department of Electrical Engineering, Faculty of Engineering, Universitas Diponegoro, Semarang, Indonesia
[4]Faculty of Technical and STEAM, HANO Technical University, Mogadishu, Somalia
[5]Department of Management Information Systems, Bucharest University of Economic Studies, Bucharest, Romania
[6]Artificial Intelligence and Robotics Laboratory, North Carolina State University, Raleigh, NC, USA
[7]School of Electrical and Information Engineering, Tianjin University, Tianjin, China

*Corresponding Author: Ferenc Erdős. Email: erdosf@sze.hu

Received: 21 February 2026; Accepted: 20 May 2026; Published: 23 July 2026

ABSTRACT: Standard retrieval-augmented generation (RAG) can perform poorly in AI for IT Operations (AIOps) settings because it is topology-blind. Basic RAG retrieves isolated, flat text snippets without enforcing structural or causal constraints, causing large language models to generate explanations that contradict the running system's actual dependency structure. To address this gap, we conducted a systematic review following PRISMA 2020, searching Scopus, IEEE Xplore, Web of Science, and Google Scholar (last searched 31 January 2026). We included empirical or systems-oriented studies applying graph-based retrieval to ground a generative model in an IT, cloud, or software-operations setting, and excluded generic RAG without an operational context and graph-only methods without a generative component. Of 139 unique records, 31 met the criteria. Because reported metrics, tasks, and hardware were too heterogeneous for pooled effect estimates, we performed a descriptive quantitative synthesis of reporting frequencies for five outcome variables (localization accuracy, text/classification scores, MTTR, retrieval/inference latency, and graph construction cost), with values harmonized to common units. The synthesis reveals that hybrid-fusion approaches have become the dominant retrieval strategy, outpacing standalone traversal in adoption for Root Cause Analysis (RCA) tasks by fusing semantic vector search with strict structural constraints. However, our evaluation matrix exposes a critical production barrier: in 73% of Service Dependency Graph (SDG)-centric studies, topology drift or streaming-update handling is not explicitly described, with many pipelines evaluated on static or periodically refreshed snapshots. We outline Event-Sourced Streaming GraphRAG (ES-GraphRAG) as a reference architecture that frames concrete design requirements for latency and drift constraints based on an event-sourced, streaming construction pattern for snapshot-consistent retrieval. The framework also incorporates strict retrieval-time governance and budget-aware traversal to help keep LLM grounding topologically accurate and compliant with incident-response Service Level Objectives (SLOs).

KEYWORDS: Knowledge graphs; retrieval-augmented generation (RAG); GraphRAG; AIOps; IT operations; root cause analysis

1 Introduction

1.1 The Rise of Generative AIOps

IT Operations (ITOps) has long relied on numerical and statistical techniques, including threshold-based monitoring, time-series anomaly detection, alert correlation, and log pattern mining, to detect and diagnose incidents [1,2]. Cloud-native platforms have changed the shape of incidents. Failures often span large microservice dependency graphs, overlap with deployment changes, and leave evidence across logs, metrics, traces, tickets, and runbooks [3,4]. Detection is rarely the hard part. The bottleneck is sensemaking: engineers must assemble a coherent explanation, check causal plausibility, and choose a safe response under time pressure [5].

This shift has driven interest in Generative AIOps, where large language models (LLMs) support root cause analysis (RCA), incident summarization, troubleshooting guidance, and postmortem drafting [6]. In practice, these assistants reduce manual sifting by synthesizing fragmented evidence and turning telemetry into readable hypotheses and candidate remediation actions [7]. Transformer architectures make this feasible at scale across large operational corpora [8].

Text-only retrieval-augmented generation (RAG) remains a common grounding pattern, but it can be a poor fit for ITOps [9,10]. Standard RAG retrieves "relevant" text chunks (e.g., log lines, manuals, tickets) and injects them into the LLM context window [9]. The result is a flat evidence set: snippets are retrieved independently, with limited enforcement of structural or causal constraints. In distributed systems, causal plausibility depends on the dependency topology (e.g., Service A calls Service B, and B depends on C). Topology is decisive for correct fault localization [11], and LLMs are prone to ungrounded outputs when context is incomplete [12], suggesting a risk of topology-inconsistent explanations when grounding is text-only. Beyond retrieval mechanics, a broader research direction articulates a bidirectional roadmap for unifying LLMs and knowledge graphs in which structured knowledge serves as an external grounding substrate that compensates for the parametric, black-box nature of LLMs [13]. For RCA, that error is costly. It can point on-call engineers to downstream components, waste investigation time, and suggest unsafe actions [3,5].

1.2 The GraphRAG Solution

Knowledge graphs use a graph-based data model to represent knowledge across application scenarios such as the large-scale integration, management, and value extraction from heterogeneous data sources [14]. Graph-based Retrieval-Augmented Generation (GraphRAG) has emerged to address the lack of structural awareness in text-only RAG [15,16]. Seminal work introduced LLM-derived entity knowledge graphs with hierarchical community summaries to enable both local and global query-focused retrieval, demonstrating substantial gains over flat-text RAG baselines [17]. Complementing these integration patterns, agentic graph-walking approaches such as Think-on-Graph treat the LLM as an interactive reasoner that performs beam search over entity-relation paths, demonstrating in open-domain question answering that iterative graph traversal can improve answer faithfulness relative to text-only retrieval [18]. Building on general GraphRAG definitions [19], we define AIOps-context GraphRAG as a system in which an operational graph serves as the primary grounding substrate for retrieval, reasoning, and evidence presentation.

Operational graphs in the literature take several forms, most commonly knowledge graphs (KGs) and service dependency graphs (SDGs) [11,20,21]. They encode entities such as services, incidents, alerts, changes, and evidence artifacts, along with typed relations (e.g., depends_on, calls, affects, caused_by, correlates_with, deployed_to) [11,20,21]. GraphRAG is not simply "RAG + a graph database". The retrieval step is graph-aware [16,19]. Instead of isolated passages, systems retrieve paths, neighborhoods, constrained subgraphs,

or graph-linked evidence bundles through traversal, graph queries, constrained propagation, or causal search [22,23]. This structure helps maintain context topology consistency and makes outputs easier to audit.

A second trend is hybrid retrieval, which combines semantic similarity search (embeddings, vector search, reranking) with graph traversal and structural constraints [16]. This hybrid design addresses a practical mapping challenge: incident text and logs are inherently unstructured, whereas operational graphs are strictly typed and structured. Vector search can locate candidate entities or relevant evidence snippets. Graph traversal then expands context along permissible dependency or causal relations. The goal is to retrieve semantically relevant evidence while still respecting operational structure.

1.3 The Systematic Gap

While Díaz-de-Arcaya et al. [24], Notaro et al. [1], and Yu et al. [25] catalog AIOps as task taxonomies (anomaly detection, alert correlation, RCA, incident triage), they typically treat the knowledge/service graph as an input and devote limited attention to how it is built or maintained from telemetry. Key engineering steps (telemetry ingestion, entity resolution, schema/ontology design, temporal provenance, incremental updates, and drift control) are often under-specified, making it difficult to explain why a retrieval strategy that appears effective offline can degrade when the graph is stale or mislinked. Conversely, GraphRAG/RAG surveys are primarily oriented toward open-domain QA [19] and only partially reflect AIOps requirements such as topology-dependent causality, non-stationary graphs, and grounding in runbooks and operational change records. Taken together, the existing review literature provides limited pipeline-level synthesis that connects telemetry-to-graph construction decisions (Construction), dependency-aware subgraph/evidence selection (Retrieval), and production constraints (Production) such as latency/availability budgets, staleness tolerance, permission/PII gating, and on-call failure modes. This review addresses that gap by systematically analyzing these layers jointly and mapping cross-layer dependencies and failure modes that influence whether a method transfers from offline evaluation to on-call use.

Unlike prior AIOps surveys that primarily organize the space by task taxonomy (e.g., anomaly detection, correlation, RCA), our contribution is a systems-oriented synthesis of the end-to-end GraphRAG pipeline used in operations. We connect (i) how to construct and maintain operational graphs from telemetry and enterprise artifacts, (ii) the retrieval operators that select topology-consistent subgraphs/evidence for generation, and (iii) the production constraints that determine whether these designs survive on-call use. By treating GraphRAG as an online system with explicit correctness, latency, drift, and governance contracts, we surface cross-layer dependencies and failure modes that are invisible in taxonomy-only reviews. This framing yields actionable design and evaluation requirements for deployable GraphRAG in real IT operations.

- First, we develop an artifact-type taxonomy of operational graphs, distinguishing Static Knowledge Graphs used for reporting and summarization from Dynamic Service Dependency Graphs used for remediation and RCA; separately, we analyze two recurring construction logics (topology-first vs. schema-first) that cut across these artifact types.
- Second, we synthesize retrieval and ranking strategies, highlighting the shift toward hybrid vector–graph fusion and differentiating structural (path/subgraph) retrieval from semantic seeding and text-rich evidence linking.
- Third, we critically assess evaluation practices and production readiness, emphasizing gaps in latency accounting, graph drift handling, and governance/PII controls.

This paper goes beyond a descriptive review to provide a quantitative meta-analysis of current performance metrics. Based on the identified gaps, it derives deployability-oriented design requirements and outlines a novel reference architecture (the Event-Sourced Streaming GraphRAG Framework) as a research agenda for future empirical work. To clarify the scope of novelty, we do not claim that event sourcing,

temporal graph views, or budget-aware serving controls are individually new techniques. Rather, our contribution is to synthesize these previously scattered systems practices into an AIOps-specific GraphRAG reference architecture and to formalize the operational contracts they must satisfy.

To avoid ambiguity, we use four analytically distinct coding axes throughout: (1) graph artifact type (KG vs. SDG), (2) construction logic (topology-first vs. schema-first), (3) task theme (Theme 1: RCA; Theme 2: summarization; Theme 3: remediation; Theme 4: correlation), and (4) retrieval operator family (traversal, query, embedding, hybrid-fusion). These axes are conceptually separable for coding and synthesis, although certain combinations may co-occur empirically within the corpus. Results are organized by RQ1–RQ3 (Construction, Retrieval, Production), while task themes are used as a cross-cutting lens when comparing design choices and evaluation practices across use cases.

This review is guided by three research questions:

RQ1: How are operational graphs constructed from heterogeneous telemetry and knowledge sources?

RQ2: What retrieval and ranking strategies distinguish GraphRAG in AIOps?

RQ3: What are the production barriers regarding latency, drift, and governance?

2 Methodology

2.1 Search Strategy and Data Sources

This review followed the PRISMA guideline for transparent, reproducible reporting of search, selection, and inclusion [26,27]. We searched Scopus, IEEE Xplore, Web of Science, and Google Scholar (noting potential indexing delays between Scopus and IEEE Xplore). The search window covered 2000–Jan 2026, and the final searches were executed in January 2026. Because GraphRAG-style grounding is recent, database-specific year filters were applied where supported to focus on contemporary work, while keeping a complete PRISMA audit trail.

We used database-specific boolean strings (full queries documented per database). In brief, each query combined:

1. GraphRAG and graph-guided retrieval terms (e.g., "GraphRAG", "graph-based RAG", "graph-aware retrieval"), OR a conjunction of
2. LLM/RAG terms (e.g., LLM, "retrieval augmented generation", GenAI, GPT) AND graph terms (e.g., "knowledge graph", "dependency graph", "service graph", "causal graph", GNN), together with
3. IT operations/AIOps terms (e.g., AIOps, ITSM, incident management, RCA, SRE, microservices, log analysis).

The initial search returned 221 records: 82 (Scopus), 40 (IEEE Xplore), 14 (Web of Science), and 85 (Google Scholar). All records were exported to a reference manager and manually cleaned. We deduplicated entries, removed non-scholarly record types (e.g., books and theses) and non-article record types (e.g., entire proceedings volumes), while retaining individual conference papers where eligible, and normalized metadata. This process yielded 139 unique records for title and abstract screening. The complete search strings for Scopus, Web of Science, and the other queried databases, together with the abstract- and full-text-screening decision logs, the PRISMA 2020 checklist, the data-extraction coding tables, and the quantitative metrics matrix, are provided in the Supplementary Materials.

2.2 Selection and Extraction

Initial screening was accelerated using Elicit, an AI research assistant, to rank papers by relevance. To mitigate potential bias from AI-assisted screening and extraction, we used a human-over-the-loop protocol:

(i) all inclusion decisions were made by the authors after full-text verification; (ii) a random 10% sample of excluded abstracts was manually audited to check for false negatives; and (iii) all AI-extracted fields and every extracted numeric value used in the quantitative synthesis were verified by the authors against the original source text/PDF. Elicit has been discussed as a research support tool for literature review workflows, and recent evaluations highlight both its utility as an accelerator and its limitations as a replacement for traditional searching [28–30]. In this review, AI assistance was used strictly for prioritization and structured extraction; all inclusion decisions and extracted fields were verified by the authors.

Studies were included if they (i) were empirical or systems-oriented research articles, (ii) applied graph-based retrieval to ground an LLM or other generative model, and (iii) addressed an IT, cloud, or software-operations setting (e.g., microservices, Kubernetes, enterprise ITSM, networking). Studies were excluded if they proposed general RAG methods without an operational context, or if they focused solely on graph neural networks without an LLM or generative component. 139 records were screened at the title/abstract stage; 46 were retained for full-text assessment, and 93 were excluded at the abstract stage. Table A1 describes the details of the title/abstract screening process.

Full-text screening of the 46 retained records yielded 31 included studies and 15 full-text exclusions, driven mainly by insufficient graph-guided grounding/retrieval at full text or domain mismatch (Fig. 1). All the 31 included studies were published after 2023.

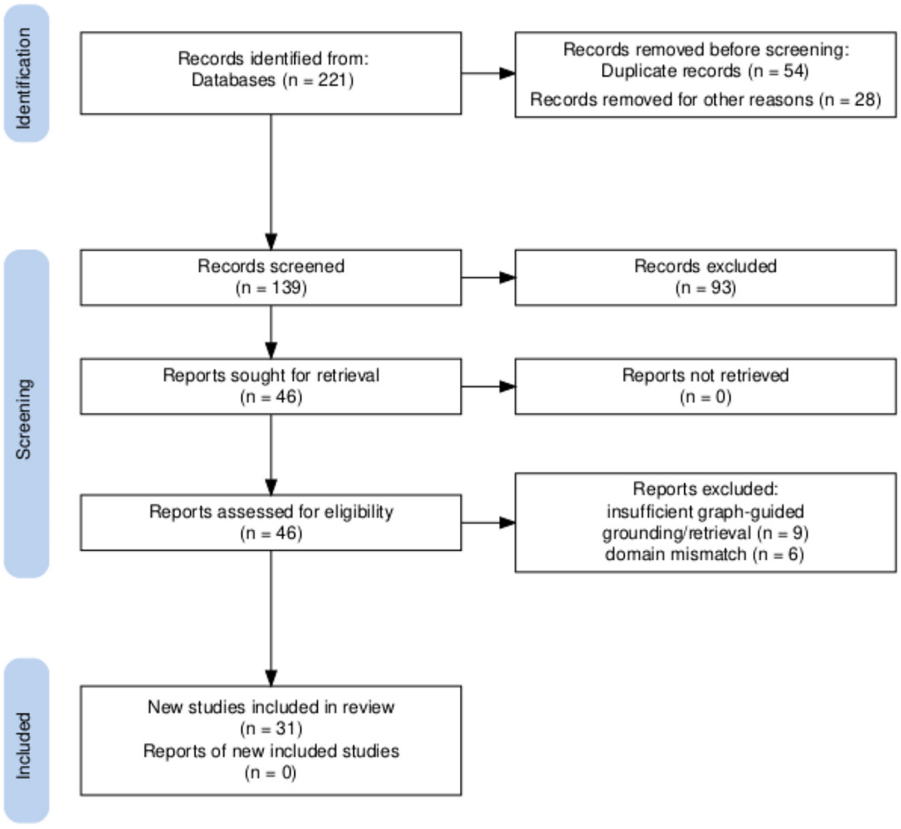

Figure 1: PRISMA flowchart.

The final data extraction matrix (our corpus dataset) comprised 20 fields (Table A2) covering scope, graph construction, retrieval strategy, generation patterns, production constraints, and evaluation design. From these verified fields, we derived a coding scheme (Section 2.3). We coded each study along four

analytically distinct coding axes (graph artifact type, construction logic, task theme, and retrieval operator family) to support systematic cross-study comparisons. Because the extraction matrix was defined at the level of published evidence, each major field was preserved with supporting quotes/tables/reasoning. Accordingly, unless a study explicitly stated that a mechanism was absent, missing information was interpreted as "not reported in the paper" rather than definitive proof of non-implementation. We used Elicit to generate the data extraction matrix. The authors then verified it row by row against the full texts to correct errors and resolve ambiguities.

2.3 Coding Scheme and Taxonomy Framework

To synthesize heterogeneous GraphRAG-for-AIOps pipelines in a reproducible way, we coded each included study (n = 31) along four analytically distinct coding axes derived from the verified extraction matrix (AIOps use case(s), Graph type & representation, Graph construction method, Graph retrieval strategy, and Hybrid retrieval & ranking). We use "analytically distinct" to indicate that these axes capture different aspects of pipeline design for coding purposes; this does not imply statistical independence in the corpus. In practice, certain combinations may co-occur more frequently (e.g., schema-first constructions in KG-oriented studies and topology-first constructions in SDG-oriented studies).

Axis A (Graph artifact type: KG vs. SDG). Knowledge graphs (KGs) denote ontology- or schema-driven graphs populated from semi-static enterprise knowledge sources (e.g., ITSM/CMDB (IT Service Management/Configuration Management Database) records and documentation). Service dependency graphs (SDGs) denote operational dependency/causal graphs constructed from runtime telemetry and topology (e.g., traces, metrics, logs, and service call relations).

Axis B (Construction logic: topology-first vs. schema-first). Topology-first construction derives nodes/edges primarily from observed dependencies or telemetry and then enriches them with semantics. Schema-first construction starts from an explicit ontology/schema and populates entities and relations from curated repositories and documents.

Axis C (Task theme: RCA/summarization/remediation/correlation). Themes were assigned based on reported AIOps use case(s) and tagged as Theme 1 (RCA: Root Cause Analysis (RCA) & Anomaly Diagnosis), Theme 2 (summarization: Incident Summarization & Postmortem Analysis), Theme 3 (remediation: Remediation Planning & Runbook Automation), and Theme 4 (correlation: Correlation & Impact Analysis). We treat anomaly diagnosis and fault localization as part of the RCA theme.

Axis D (Retrieval operator family: traversal/query/embedding/hybrid-fusion). Traversal covers constrained graph search and bounded expansions (e.g., k-hop neighborhoods, random walks). Query covers explicit graph queries and pattern matching. Embedding covers node/edge representations used for similarity-based selection. Hybrid-fusion covers an explicit combination of graph retrieval with vector/embedding retrieval and/or reranking.

Task theme is multi-label: a single study may contribute to multiple operational tasks, so theme counts can exceed n = 31. The other axes were coded with a single dominant label per study; where papers did not report sufficient detail to categorize retrieval, they were coded as unclear for that axis.

Fig. 2 summarizes the coding outcomes for the 31 included studies under the four analytically distinct coding axes. The counts in Fig. 2 are used as the quantitative backbone for Section 3, enabling cross-tabulation of design choices against evaluation and reporting gaps. A study-level coding sheet (Table A3) reports the per-study assignments under the four-axis framework.

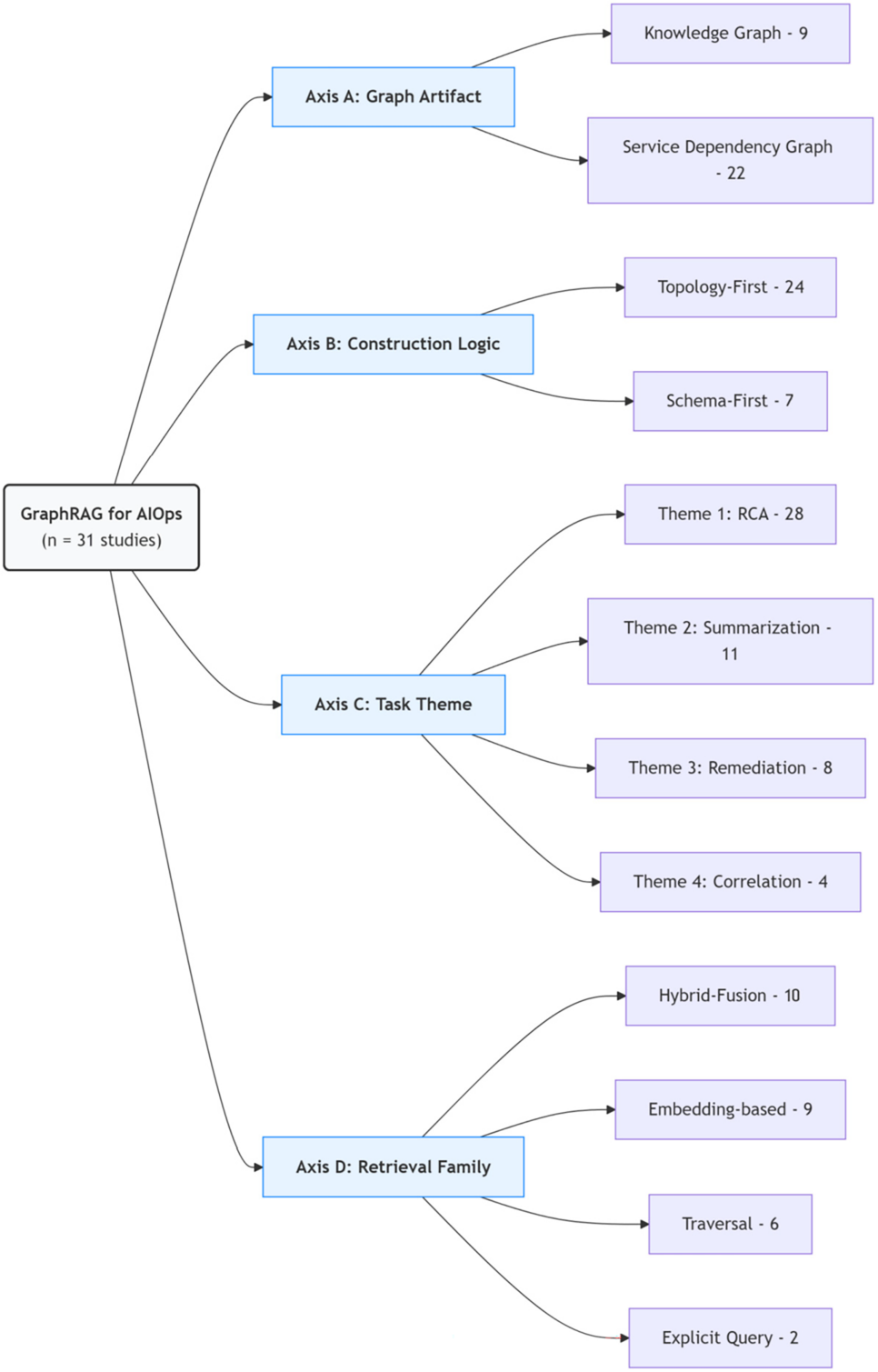

Figure 2: Taxonomy of GraphRAG in AIOps across four analytically distinct coding axes: (**A**) graph artifact type, (**B**) construction logic, (**C**) task theme, and (**D**) retrieval operator family.

Fig. 3 summarizes the end-to-end GraphRAG-for-AIOps pipeline and shows where each coding axis attaches. "Construction" covers telemetry preprocessing and the rules that map logs/metrics/traces/ITSM context into a persistent graph artifact (KG or SDG). "Retrieval" selects and constrains evidence (subgraphs, paths, neighborhoods, or hybrid fused candidates) from that artifact, and "Generation" produces operator-facing outputs that drive RCA, summarization, remediation, or correlation. The dashed boundary highlights that serving-time constraints (latency/availability budgets, drift/staleness, and governance/PII gating) act on retrieval and generation, which motivates coding pipelines by artifact type, construction logic, operator family, and task theme.

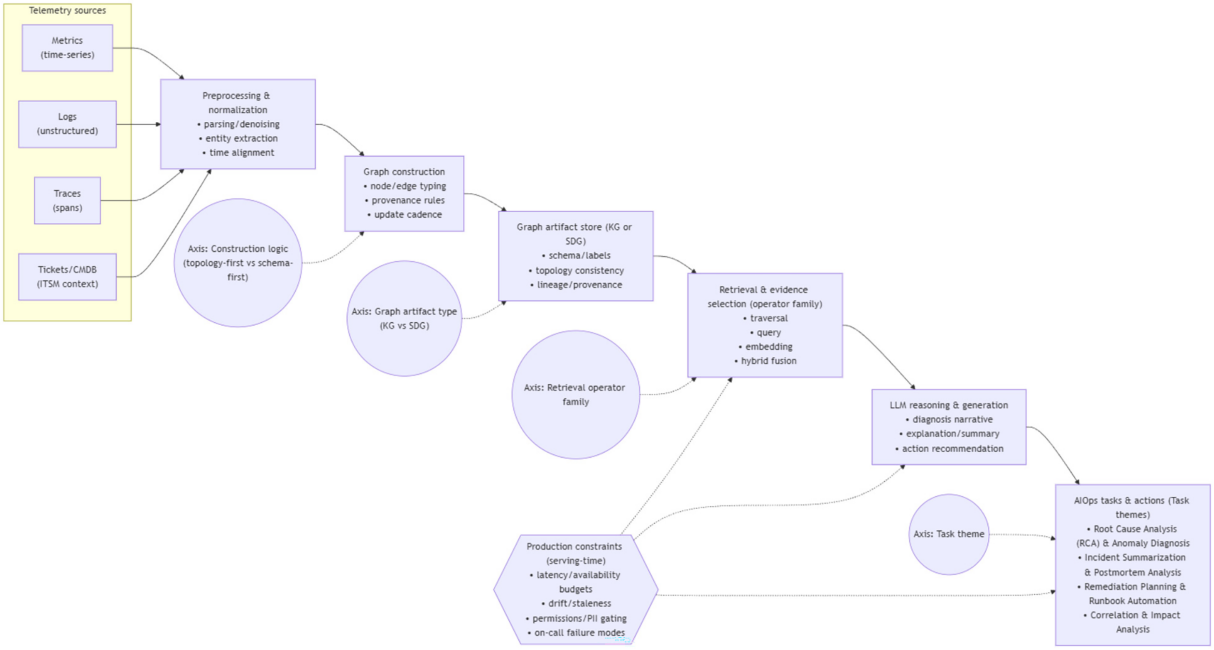

Figure 3: End-to-end GraphRAG-for-AIOps pipeline and where each coding axis applies.

2.4 Quantitative Synthesis of Reported Metrics

To complement the qualitative taxonomy and rigorously address the deployment viability of current GraphRAG systems, we conducted a targeted quantitative meta-analysis of the performance metrics reported across the 31 included studies. A conventional effect-size meta-analysis was infeasible for this corpus because the included studies do not evaluate a common intervention–outcome pair under comparable conditions. The papers span heterogeneous operational tasks (e.g., RCA, incident summarization, remediation, correlation) and report non-commensurate metrics (ranking metrics, text/classification scores, success rates, or case-study outcomes), often on different datasets, workloads, and system architectures. Reported performance is further confounded by hardware and deployment assumptions (e.g., model size, GPU/CPU configuration, graph store/indexing choices, caching, and retrieval depth), and many studies omit the summary statistics needed for pooling (e.g., variance/CI, repeated-trial design, consistent baselines). Therefore, we adopt a quantitative synthesis of reported metrics focused on (i) reporting frequency and (ii) normalized ranges (e.g., latency unit harmonization), rather than attempting to compute pooled effect estimates.

Given the severe heterogeneity in reporting formats, in which critical metrics are often embedded in dense text, unstructured tables, or supplementary appendices, we utilized an AI-assisted full-text data extraction pipeline. Specifically, we employed Elicit's reasoning-based extraction features. Although recent

evaluations report mixed performance for AI-assisted screening and search (notably reduced recall vs. expert searching [29]), Elicit has been shown to be useful as an accelerator when paired with human verification [28–30]. We therefore treat all AI-extracted fields as candidate values that require author verification. We engineered strict, prompt-based extraction criteria targeting five specific operational variables: (1) Culprit Localization Accuracy (e.g., Hit@k, Recall@k, MRR), (2) Text and Classification Metrics (e.g., Precision, F1-score), (3) Operational Outcomes (e.g., MTTR reduction), (4) Retrieval and Inference Latency, and (5) Graph Construction Cost. To ensure absolute data integrity, every AI-extracted numeric value and its surrounding context were manually verified by the authors against the original source PDFs. This verification step ensured that the extracted metrics corresponded explicitly to the authors' proposed frameworks rather than baseline comparative models. The verified dataset was subsequently normalized to common units (e.g., converting all latency values to seconds) to compute reporting frequency distributions and latency variance, yielding the empirical dataset utilized in Section 3 to evaluate the field's production readiness.

2.5 Limitations

This review is subject to several limitations related to evidence availability and reproducibility. Selection and reporting bias is unavoidable: studies that report serving-time latency constitute a non-random subset of the literature, and even when latency is reported, values are not directly comparable across papers due to differences in hardware, software stacks, caching/batching, and most importantly, measurement boundaries (e.g., retrieval-only vs. retrieval + generation, warm vs. cold cache, single-query latency vs. throughput). For this reason, our quantitative synthesis treats latency primarily as a reported deployment-readiness signal rather than a pooled benchmark.

Additionally, the corpus includes multiple 2025–2026 preprints, whose methods and results may change across versions or differ from later peer-reviewed publications. Findings derived from preprints should therefore be interpreted as provisional. Finally, the synthesis relies on categorical coding; we mitigate subjectivity through a predefined coding scheme and documented decision rules.

3 Results

3.1 Graph Construction and Representation in AIOps (RQ1)

Graph construction is not a preprocessing detail in GraphRAG-for-AIOps: it defines the admissible retrieval space for topology-constrained expansion (which nodes/edges may be traversed), the operational semantics of "evidence" (what a retrieved subgraph denotes), and the dominant failure modes (staleness, mis-linking, non-auditable edges). Across the corpus (n = 31), service dependency graphs and related SDG-style artifacts dominate (22/31), while knowledge graphs appear in 9/31 studies (Fig. 2). Construction is predominantly topology-first (24/31) rather than schema-first (7/31), so graph fidelity—not the LLM alone—sets the practical upper bound on what downstream retrieval and generation can claim.

Topology-first construction treats the service dependency graph as an observed backbone derived from distributed traces, service maps, configuration state, or causal structure learning, then attaches metrics, logs, and alerts as node or edge attributes [31]. This tends to preserve runtime dependency semantics for fault localization because edges encode execution or dependency relations, but it can underrepresent first-class operational objects (incidents, changes, runbooks, tickets) needed for cross-artifact joins. In topology-first pipelines, the key engineering contract is identity plus edge semantics: if trace-to-service mapping is ambiguous or if causal edges are inferred without stated assumptions, traversal returns plausible-looking but non-auditable "evidence" [32,33].

Schema-first construction inverts priorities by defining a typed entity–relation vocabulary (services, incidents, alerts, changes, deployments, runbooks, evidence artifacts) and populating it from heterogeneous operational text and telemetry, often via LLM-mediated extraction or induction. This supports cross-artifact queries ("incident → affected service → preceding change → relevant runbook") because joins are explicit typed edges rather than implicit associations. However, schema-first graphs shift the failure mode from missing topology to fabricated or weakly supported links; without explicit provenance (source, extractor, confidence) and verification, the graph can propagate unsupported relations and increase the risk of ungrounded generated explanations [34]. Several studies gesture at mitigations (rule-constrained linking, human verification, or curated schemas), but these mechanisms are rarely specified as enforceable contracts [7,20].

Operationally, "graph quality" is multi-dimensional. Topology-first pipelines require controlled semantic augmentation without corrupting dependency meaning. Schema-first pipelines require systematic provenance verification and consistency enforcement to prevent semantic richness from degenerating into fabricated edges. A deployable GraphRAG pipeline should treat construction as an engineered contract, including admissible entity/edge types, provenance guarantees, and consistency-enforcement rules that reduce the risk of non-falsifiable explanations.

3.2 Retrieval Strategies and LLM Integration (RQ2)

Across the corpus, retrieval is implemented via four operator families (Fig. 2): hybrid-fusion (10/31) is the most frequently reported operator family in the corpus, followed by embedding-based retrieval (9/31), traversal (6/31), and explicit query/pattern matching (2/31); 4/31 papers do not report sufficient detail to categorize retrieval. Operator choice is not cosmetic: the dominant task themes are RCA (28 studies) and incident summarization (11 studies), and these tasks impose different correctness contracts (topology-consistent blame assignment vs. coverage-oriented context building). Fig. 4 cross-tabulates the different task themes against the dominant retrieval operator family. RCA dominates the study set and spans all operator families, with the largest mass concentrated in embedding-based and hybrid-fusion retrieval. Summarization tasks disproportionately co-occur with hybrid-fusion pipelines, consistent with designs that combine graph-topology evidence with semantic retrieval and reranking before generation. Remediation and correlation appear less frequently overall and rarely use explicit query-style operators. Instead, they are implemented primarily through traversal or embedding/hybrid pipelines. Because the task theme axis is multi-label, the row totals in Fig. 4 may exceed N = 31.

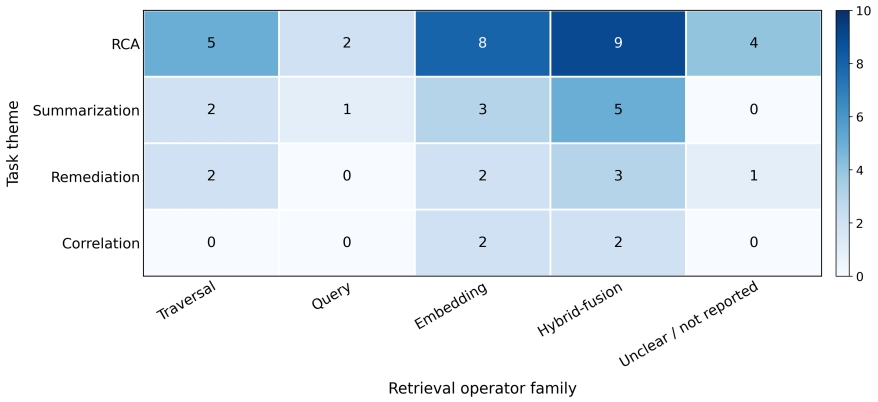

Figure 4: Cross-tabulation heatmap of task theme vs. retrieval operator family (counts).

Traversal-family retrieval is emphasized when the explanation object must be topologically admissible. For RCA, path tracing implements topology-constrained traversal because it mirrors fault propagation over dependency structure: traversal is not merely a retrieval primitive; it is the candidate explanation object. Bounded BFS expansion, iterative backward search, and other constrained traversal operators recur in RCA pipelines because they reconstruct responsibility chains rather than collecting semantically similar evidence [22,23,35,36].

For summarization and reporting, traversal is often used as neighborhood aggregation rather than a single-path explanation. In these workloads, k-hop neighborhood expansion and entity-centric aggregation retrieve state, evidence snippets, and related objects around an incident locus, even when causality is uncertain or multi-factorial [21,37]. This increases coverage and reduces sensitivity to missing causal edges, but it weakens causal discipline: neighborhood context can be relevant without being diagnostic, so aggregation needs explicit selection and redundancy control to avoid prompt inflation.

Hybrid-fusion and embedding-family approaches introduce an arbitration gap that remains underspecified. When vector-based candidate generation conflicts with topology constraints, many studies do not fully specify the fusion operator (e.g., whether topology prunes candidates before scoring, whether evidence is merged at score level, or whether candidates are jointly reranked), leaving the end-to-end retrieval function underspecified and hard to reproduce [21]. GRACE is a concrete illustration of the reporting gap: despite centering a service dependency graph and graph RL for recovery, it does not specify a concrete graph-retrieval operator family (traversal/query/embedding) or a fusion policy that would make evidence selection reproducible [38].

When symptoms originate as monitoring questions rather than free text, retrieval planning reduces to text-to-query compilation (e.g., LLM synthesis of PromQL to fetch KPI slices that seed subsequent expansion), not embedding-only search [39]. This query-family pattern is rare in the corpus (2/31) but is operationally crucial because it exposes a distinct failure mode: incorrect compilation yields incorrect seeds, and downstream graph retrieval can only be as correct as its initial telemetry slice.

3.3 Evaluation Methodologies and Deployment Viability (RQ3)

RQ3 examines whether GraphRAG-for-AIOps pipelines are specified and evaluated in ways that make them deployable under real operational constraints. In production, retrieval and generation must operate under hard latency and cost budgets, must remain robust to drift and staleness in telemetry and topology, and must satisfy governance requirements (e.g., access control, privacy constraints, and auditability of evidence and actions) [24]. We therefore analyze reporting and evaluation practices. Because production risks depend on what the graph represents and how it is maintained, we stratify the reporting gaps by graph artifact type (KG vs. SDG), as summarized in Fig. 5. The figure should be interpreted as a paper-level reporting-gap matrix: unless a study explicitly stated that a mechanism was absent, red cells indicate mechanisms that were not reported or not explicitly described in the publication, rather than mechanisms proven to be absent in the implemented system. The most prominent paper-level reporting gaps concern graph drift handling (20/31), latency/cost bounding (18/31), and entity resolution (12/31).

Fig. 6 reports normalized percentages (rather than raw counts) to enable a fair comparison between the KG and SDG subsets. Our analysis shows that SDG-centric studies are more likely to omit drift/staleness treatment (16/22; 73%) than KG-centric studies (4/9; 44%), despite SDG pipelines being tightly coupled to fast-changing telemetry and service topology. In contrast, SDG studies are less likely to omit latency/cost-bounding mechanisms (11/22; 50%) than KG studies (7/9; 78%), consistent with SDG pipelines being framed as online decision loops in which retrieval depth, candidate set size, or caching must be controlled. Dataset type disclosure is also stronger in SDG studies (2/22 omitted; 9%) than in KG studies (3/9 omitted; 33%).

Overall, the dominant production-readiness gap for SDG pipelines is under-reporting of drift/staleness, whereas KG pipelines more often under-report serving-time constraints and basic dataset provenance needed to interpret operational claims.

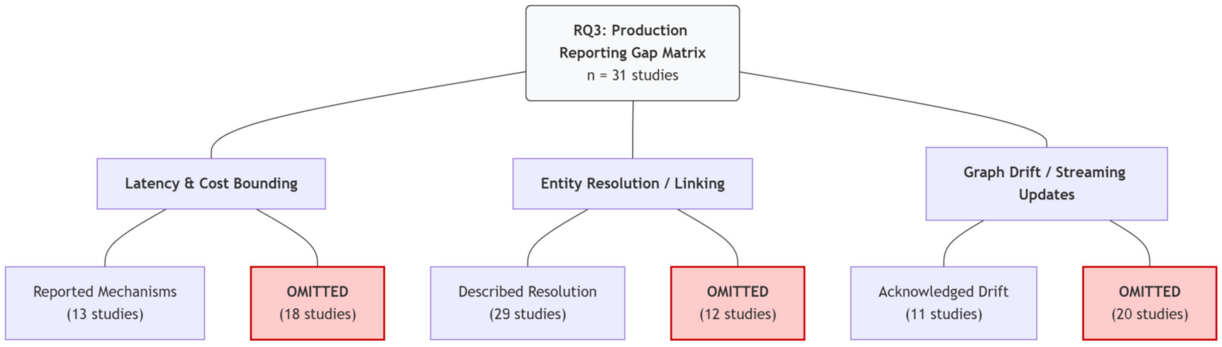

Figure 5: Paper-level reporting gap matrix highlighting critical production constraints that were not reported or not explicitly described in the publications across the 31 reviewed studies.

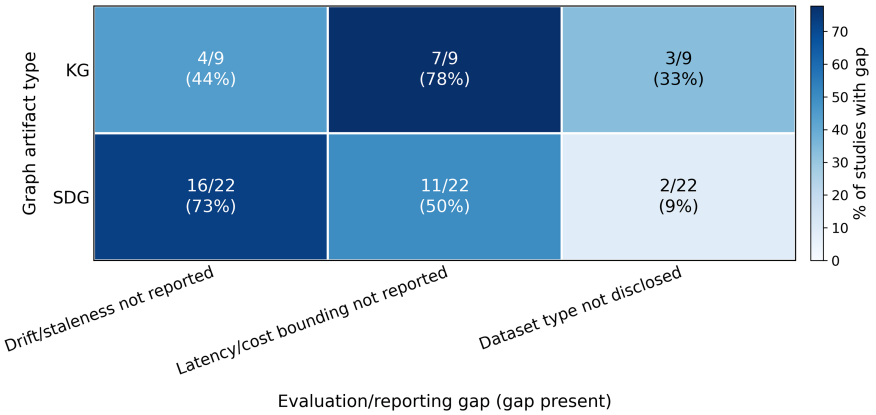

Figure 6: Cross-tabulation heatmap of evaluation/reporting gaps by graph artifact type.

3.3.1 The Metric–Task Alignment Gap

Task theme strongly conditions what constitutes a valid evaluation. In RCA-oriented studies (n = 28), only 14 explicitly report culprit-localization or ranking-style metrics (e.g., Hit@k/Recall@k/MRR or equivalent "correct root cause surfaced early" measures). A smaller subset (3) reports end-to-end operational outcomes such as MTTR reduction, and a minority (2) includes text-centric metrics even when the claim is a causal diagnosis. The evaluation gap is therefore not "insufficient metrics" in general; it is mis-specification of the target: many papers optimize what they can easily measure rather than what RCA requires.

Summarization-oriented workflows (Theme 2; n = 11) show a different failure mode. Only 1 explicitly reports text-quality or faithfulness-style metrics. In contrast, others rely on indirect proxies (classification success, runtime success rates, or high-level productivity claims) that do not test whether the narrative is grounded in retrieved evidence or improves operator actions [40]. This creates an evaluation blind spot: fluent summaries can be produced from weakly relevant context without being faithful to the underlying incident evidence.

To ensure rigorous data extraction and to quantify this metric-task alignment gap, we performed a meta-analysis of the evaluation metrics across all 31 included studies. As illustrated in Fig. 7, the results reveal a severe misalignment in evaluation standards. While 14 out of the 31 studies (45.2%) report standard text-centric or classification metrics such as Precision and F1-score, only 5 studies report true operational outcomes such as Mean Time To Repair (MTTR) reduction. Even more critically, the computational cost of constructing these knowledge graphs is almost universally ignored, being reported by only a single study. This sparsity supports our claim that current evaluations often emphasize offline metrics over operational constraints.

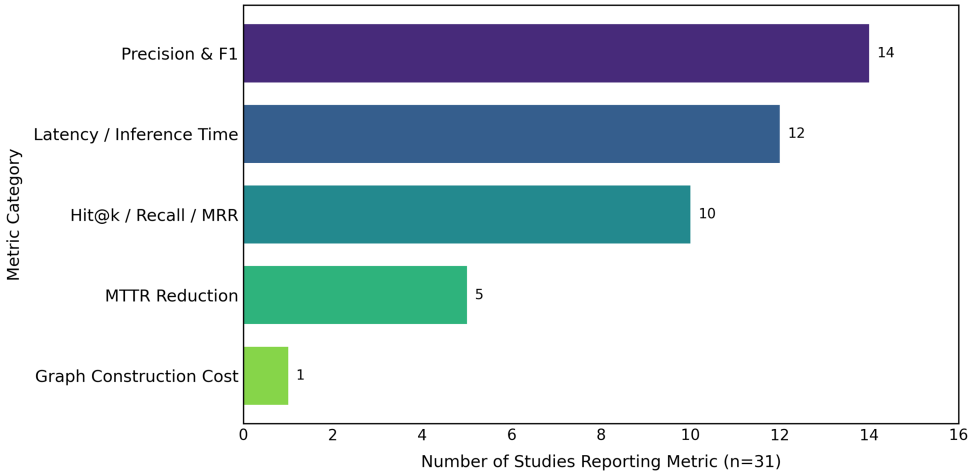

Figure 7: Reporting frequency of key evaluation metrics across the 31 studies.

RCA is fundamentally a ranking and evidence-faithfulness problem. Minimally coherent proxies are retrieval/localization metrics that test whether the true culprit is surfaced early (Recall/Hit@k, MRR, Precision@k) and whether the predicted path/subgraph aligns with the dependency mechanism being claimed [31,33,35,36,41,42]. Text-overlap metrics are poorly aligned with RCA because they reward fluency and template similarity rather than causal correctness, and they do not detect when a model "explains" with evidence that is irrelevant or inconsistent with the retrieved subgraph.

3.3.2 Data Reproducibility and the "Private Data" Problem

Dataset disclosure is uneven. Based on what studies explicitly report, 11 use public or widely shareable benchmarks, 13 rely on private/proprietary operational data, and 7 do not clearly state the dataset type. This imbalance is not uniform across artifact type: KG-oriented papers are disproportionately under-specified (5/9 do not report dataset type, and only 1/9 use public data), whereas SDG-oriented work is split between public (10/22) and private (10/22) datasets.

The reproducibility penalty is also theme-skewed. Summarization-theme studies rarely rely on public incident corpora (only 1/11 use public data; the remainder are private or not reported), which limits cross-paper comparability even when similar claims are made about LLM grounding or operator productivity. A small subset of results anchors public or widely shareable microservice benchmarks and curated datasets, enabling methodological comparison across studies [31,32,43]. Many others rely on proprietary telemetry and internal incident corpora, often without publishing topology, logging policies, incident taxonomy, or labeling processes that define "ground truth" in AIOps settings [6,7,44–46].

GraphRAG performance is not a model-only property; it is coupled to organization-specific structure. Service graphs encode local dependency conventions, observability coverage gaps, naming schemes, and change cadences, so reported gains can be artifacts of the specific topology under test rather than evidence of portable improvements [7,46]. When data are private, the only credible substitute is strict protocol disclosure: baselines that isolate the graph contribution, ablations separating construction from retrieval, and explicit ground-truth definitions for culprit, impact, and remediation success.

3.3.3 Production Barriers: Latency, Drift, and Governance

Generic scalability claims do not establish deployment viability. It is constrained by whether the pipeline can operate within an incident-response loop without violating SLO latency, corrupting evidence during topology changes, or leaking restricted data. These constraints interact with artifact type and task theme: SDG-centered RCA pipelines are most sensitive to hop/fanout blow-up and drift (because traversal is the explanation object), while KG-centered summarization/remediation pipelines are most sensitive to governance (because retrieval pulls heterogeneous, often sensitive, operational artifacts into the prompt). Three constraints define deployment viability: latency, drift, and governance.

Latency is often reported as online inference time, which is easy to measure but insufficient for systems planning. The true cost envelope decomposes into (i) offline indexing/build cost (graph construction, embedding computation, index refresh) and (ii) online retrieval plus generation latency per query. Reporting only per-query runtime can obscure major cost drivers, including re-indexing and graph refresh under ongoing change, which can dominate the incident-response budget. Mitigations such as bounded topology expansion, top-k candidate control, caching, and progressive retrieval should be reported as enforceable mechanisms, not discussed as vague "efficiency" claims [7,22].

Beyond accuracy, bounded retrieval time is arguably the most critical constraint in AIOps [3,5]. Our quantitative analysis of the 12 studies that reported retrieval or inference latency reveals the severe operational limitations of current static GraphRAG approaches. Fig. 8 shows the distribution of these reported latencies on a logarithmic scale, plotted against standard incident response thresholds. The data exhibits extreme variance, ranging from 11.8 ms to over 330 s (5.5 min). While automated triage systems might tolerate latencies up to 1 min under typical SRE incident-response budgets [47], interactive incident response (such as ChatOps or dynamic UI dashboards) requires sub-second retrieval to maintain operator flow during active outages, consistent with established response-time guidance from human–computer interaction [48]. Both thresholds are reinforced by empirical studies showing that on-call sensemaking is time-pressured and dominated by investigation cost rather than detection [3,5]. The empirical evidence shows that several state-of-the-art implementations (e.g., dense KG traversal and LLM-based reasoning agents) exceed 4 s, directly demonstrating the "Static Graph Fallacy". Unbounded graph traversal during an active incident violates strict time-to-resolution budgets, necessitating a shift toward latency-aware architectures.

Graph drift is the second failure mode: many pipelines assume a stable dependency structure while operating in environments where microservices autoscale, deployments reshape call graphs, and observability coverage changes continuously. Without explicit continuous update mechanisms, versioning, and drift-consistency enforcement, "correct" explanations decay into stale narratives that reference a topology that no longer exists [21,49,50]. Drift is not an edge case; it is the default operating condition in microservice platforms [3,4], so drift handling must be treated as part of the retrieval contract.

Governance is the third constraint and is systematically under-specified: deployable GraphRAG requires retrieval-time access control, not post-generation redaction. If the retrieval layer can inject restricted tickets, user identifiers, or sensitive logs into the prompt, policy is violated even when the final text is sanitized.

Deployment-grade pipelines therefore require retrieval-time ACL (Access Control List) enforcement and PII-aware filtering with auditable provenance trails. Yet only a minority of studies treat this as a first-class system requirement.

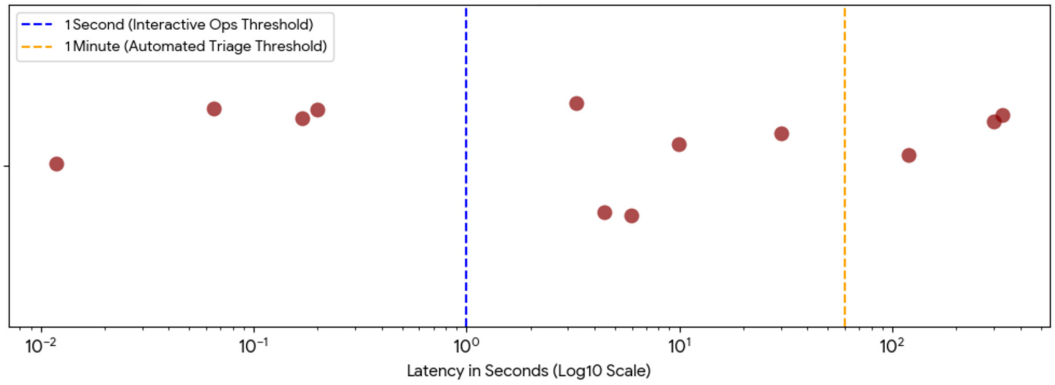

Figure 8: Log-scale distribution of reported retrieval and inference latencies across 12 GraphRAG AIOps studies, mapped against operational SLA thresholds.

4 Implications for Deployable GraphRAG in AIOps: Design Requirements and a Reference Architecture: The Event-Sourced GraphRAG Framework

Our systematic review of 31 studies reveals a fundamental disconnect between the dynamic nature of IT operations and the static graph architectures currently proposed in the literature. While 22 of the 31 included studies rely on Service Dependency Graphs (SDGs) to ground Large Language Models (LLMs), the majority do not provide explicit mechanisms or evaluation evidence for continuous topology drift handling and strict serving-time latency constraints in production environments.

As a conceptual scope note, we emphasize that the ES-GraphRAG description in this section is intended as a reference design, operationalizing the review findings into explicit system contracts (time, budget, and governance). It is not presented as a validated system contribution, and we therefore avoid claims of empirical latency or retrieval-quality improvements. Instead, we specify evaluation criteria and measurement boundaries that future prototypes should report to make such claims testable.

To address these systemic gaps, we translate these findings into deployability-oriented design requirements and outline Event-Sourced Streaming GraphRAG (ES-GraphRAG) as a reference architecture. This framework enables snapshot-consistent retrieval under standard event-sourcing and event-time processing assumptions (e.g., an append-only, complete event log; well-defined measurement boundaries; and bounded out-of-order arrival handled via watermarks [51]), making the retrieved subgraph incident time auditable and replayable.

4.1 Critical Gap Analysis: The Static Graph Fallacy

The "Static Graph Fallacy" is the assumption that a graph G_{stored} in the retrieval index is an adequate approximation of the system state $G_{runtime}$. Our synthesis suggests this assumption often breaks down along three dimensions that are frequently under-specified or under-evaluated:

1. Topology Drift and Staleness: In microservice environments, dependencies are rewritten continuously by autoscaling and deployments. Yet, our review found that only 11/31 studies explicitly describe streaming updates or drift-handling mechanisms. In the remaining studies, update cadence and drift

controls are unclear or not reported, which increases the risk that retrieval operates on stale topology during incidents.
2. Unbounded Latency: RCA is a time-critical task, yet 15/31 studies do not report enforceable latency or cost-bounding mechanisms. Without explicit budgets, graph traversal (especially hybrid-fusion approaches used in 10/31 studies) risks a combinatorial explosion, violating incident response Service Level Objectives (SLOs).
3. Identity Fragmentation: Telemetry sources often use divergent identifiers for the same service (e.g., distinct keys in traces vs. CMDBs). We found that 12/31 studies omit entity resolution entirely. Without a canonicalization layer, the graph accumulates duplicate nodes, forcing the LLM to hallucinate connections between disconnected aliases.

4.2 Formal Framework: Event-Sourced Streaming GraphRAG

To resolve the drift and latency gaps identified above, we define the ES-GraphRAG architecture. Unlike static approaches, this framework treats the graph as a materialized view of an immutable event log.

4.2.1 Problem Formulation

We define the operational graph at time t as $G_t = (V_t, E_t)$. We define Topology Drift, Δ, as the divergence between the stored graph and the runtime topology at the moment of an incident t_{inc}:

$$\Delta\left(G_{stored}, G_{runtime}\right) = \frac{\left|\left(E_{stored} \backslash E_{runtime}\right) \cup \left(E_{runtime} \backslash E_{stored}\right)\right|}{\left|E_{runtime}\right|}$$

The objective is to minimize retrieval error such that $\Delta \leq \epsilon$, where ϵ is the consistency tolerance, subject to a retrieval budget B_{total}.

4.2.2 Architectural Components

The system pipeline consists of three stages designed to replace the "batch dump" methods observed in the literature:

1. Ingestion Layer (The Event Log): Telemetry is ingested as an append-only stream of immutable events $S = \{e_1, e_2, \ldots, e_n\}$, where each event e_i contains at minimum a type, an event-time timestamp e_i.time, a payload, and provenance metadata (source system, extractor, confidence, ACL labels). Unlike batch pipelines, this layer is designed for continuous ingestion of topology and evidence changes (e.g., ServiceDeployment, AlertFiring) into a streaming backbone (e.g., Kafka). The log is the system of record: materialized graphs and indices are derived artifacts, enabling reproducibility, auditability, and offline experimentation by replay.
2. Projection Layer (Time-Travel Graph):
   A projection function P maps the event stream to a Temporal Property Graph. Every edge E_{uv} carries a validity window $[t_{start}, t_{end}]$, following the temporal property graph model and built-in versioning patterns established in recent temporal graph database research [52].

   $$G_t = P\left(S_{0..t}\right) = P\left(S_{0..t-1}\right) \oplus e_t$$

   Here, $\oplus$ denotes the incremental update (e.g., expiring an edge). This allows the system to reconstruct the exact topology that existed at t_{inc}.
3. Retrieval Layer (Snapshot-Consistent Search):

We introduce a Budget-Aware Retrieval Operator $R(q, t_{inc}, B_{total})$. This operator executes traversals only across edges valid at t_{inc}, ensuring causal consistency. It solves the following optimization problem:

$$\max_{G'_{sub} \subseteq G_{t_{inc}}} \text{Info}\left(G'_{sub} \mid q\right) \text{ subject to } \text{Cost}\left(G'_{sub}\right) \leq B_{total}$$

This explicitly addresses the latency gap by pruning low-value traversals before the cost budget B_{total} is exceeded.

4.2.3 System Execution Flow

Fig. 9 illustrates the end-to-end operational flow of the Event-Sourced Streaming GraphRAG (ES-GraphRAG) architecture. This sequence is explicitly designed to resolve the production barriers of topology drift and unbounded latency identified in our systematic review.

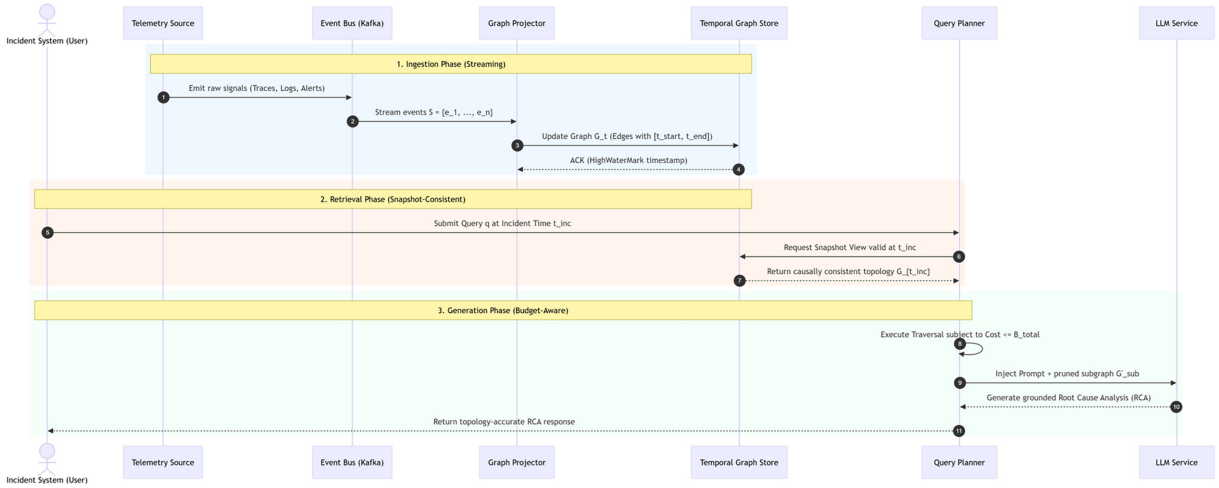

Figure 9: System sequence diagram of the ES-GraphRAG architecture.

The execution is divided into three asynchronous phases:

1. The Ingestion Phase (Streaming): Unlike conventional GraphRAG pipelines that rely on periodic batch dumps, the ES-GraphRAG framework ingests continuous telemetry (e.g., traces, logs, and deployment states) through an event bus such as Kafka. The Graph Projector converts these raw signals into an immutable event stream and updates the Temporal Graph Store. By appending a HighWater-Mark timestamp to every structural change, the system maintains a fully versioned history of the operational environment.
2. The Retrieval Phase (Snapshot-Consistent): When an anomaly occurs, an automated alert is raised or a user submits a query, each carrying the precise incident timestamp, t_{inc}. Rather than querying the current state of the database, which may have already been altered by autoscaling or automated rollbacks, the Query Planner requests a snapshot of the topology exactly as it existed at t_{inc}. This "time-travel" capability directly neutralizes the static graph fallacy, ensuring that the retrieved subgraph $G_{t_{inc}}$ represents the true operational reality at the moment of failure, preventing the generation of stale or hallucinated narratives.

3. The Generation Phase (Budget-Aware): Once the causally consistent topology is retrieved, the Query Planner executes a bounded traversal. Our review noted that 48% of the analyzed studies lack enforceable latency or cost bounding, which can lead to a combinatorial explosion during complex incidents. To prevent this, traversal is constrained by a strict resource budget B_{total}, evaluating the diagnostic value of each edge against its computational cost. The pruned, high-value subgraph G'_{sub} is then injected into the Large Language Model (LLM) context window, yielding a root cause analysis that is both topologically accurate and delivered within the incident-response Service-Level Objective (SLO).

4.2.4 Algorithm: Snapshot-Consistent Retrieval

The following pseudo-algorithm (Algorithm 1) implements the logic required to enforce the contracts missing from the surveyed literature (drift control and latency bounding).

Algorithm 1: Budget-aware snapshot-consistent subgraph retrieval

```
Input: Query q, Incident Time t_inc, Budget B
Output: Context C_final
Entities <- EntityResolver.Resolve(q)          // Solves Identity Gap
View_V <- GraphStore.GetView(time = t_inc)     // Solves Drift Gap
Queue <- PriorityQueue(Entities)
Subgraph <- EmptyGraph()
CurrentCost <- 0
While Queue not Empty AND CurrentCost < B:
    Node <- Queue.Pop()
    Neighbors <- View_V.GetNeighbors(Node)     // Validated at t_inc
    For each n in Neighbors:
        Cost_n <- EstimateCost(n)
        If CurrentCost + Cost_n > B: Break     // Enforces Latency Budget
        Subgraph.AddEdge(Node, n)
        CurrentCost + = Cost_n
        Queue.Push(n)
Return Serialize(Subgraph)
```

4.3 Implementation Requirements: Canonicalization and Governance

For the ES-GraphRAG architecture to function in a production setting, two auxiliary services are required to address the remaining gaps identified in our review: identity fragmentation and governance.

- Probabilistic Canonicalization Service. To address the identity gap found in 39% of studies, the ingestion layer must include a canonicalization service. This service assigns stable Canonical Identifiers (CIDs) to entities across heterogeneous sources (e.g., mapping a Kubernetes Pod ID in logs to a Service ID in the CMDB). We suggest a two-stage resolver: deterministic joining on unique keys followed by probabilistic linkage based on neighborhood overlap. This two-stage pattern—combining deterministic matching on stable keys with probabilistic matching that exploits relational evidence—aligns with the broader end-to-end entity resolution literature for Big Data, where neighborhood-based similarity is widely used to resolve entities whose attribute values alone are insufficiently discriminative under high Volume, Velocity, and Variety conditions [53].
- Retrieval-Time Access Control (ACL). Governance is systematically under-specified in the literature, with strictly enforced ACLs nearly absent. In the ES-GraphRAG framework, governance must be enforced at the retrieval layer. Every node and edge in the temporal graph must carry security labels.

The retrieval operator R filters the adjacency list based on the requesting principal's permissions before adding nodes to the subgraph. This ensures that sensitive artifacts (e.g., PII in logs) are never injected into the LLM context window, shifting governance from post-generation redaction to a strict retrieval invariant [54].

4.4 Summary of Contribution

This proposed architecture shifts the paradigm from "best-effort" text retrieval to engineered, contract-aware retrieval for operations. By replacing static graphs with event-sourced projections, ES-GraphRAG mitigates topology drift by supporting retrieval against a temporally versioned graph. The remaining staleness is explicitly bounded by watermark lag and by the completeness of the event log. By making budgets first-class (rather than implicit hop limits), the reference design clarifies where and how serving-time cost/latency constraints should be enforced and measured in future systems. Overall, ES-GraphRAG provides a blueprint for deploying GraphRAG in high-velocity AIOps settings with explicit assumptions about time, drift, and serving-time budgets. ES-GraphRAG should therefore be read as a deployability-oriented reference architecture that integrates established systems patterns into a GraphRAG-for-AIOps setting, rather than as a claim of novelty for each infrastructural mechanism in isolation.

5 Conclusion

Our systematic meta-analysis of 31 recent studies exposes a severe "Evaluation Gap" that actively undermines the deployment viability of GraphRAG in IT Operations. Although the current literature effectively demonstrates the theoretical utility of grounding large language models in operational knowledge, evaluation methodologies remain fundamentally misaligned with the high-velocity realities of production environments. The empirical data are unequivocal: while 41.9% of the reviewed pipelines optimize for offline text-centric or classification metrics, such as Precision and F1-score, only 5 studies report genuine operational outcomes, such as Mean Time To Repair (MTTR) reduction. Most alarmingly, the reported retrieval and inference latencies exhibit substantial variance across studies, ranging from 11.8 ms to 330 s (5.5 min); values at the upper end of this range are operationally infeasible for interactive incident-response workflows.

These quantitative findings suggest a shift in emphasis for future work: future systems research should shift away from static, schema-first batch implementations toward topology-first, event-sourced streaming architectures. The prevalent reliance on periodic graph dumps reinforces the "Static Graph Fallacy", wherein a periodically refreshed retrieval index may not capture the rapidly evolving runtime topology of modern microservices. Because continuous topology drift is the default operating condition in cloud-native platforms, executing unbounded graph traversals over these stale representations is inherently unsafe. Such static architectures not only feed large language models with decoupled, hallucinated dependencies, but also, through their unrestrained traversal mechanisms, actively violate the strict incident-response Service Level Objectives (SLOs) required during critical outages.

To bridge this critical chasm between academic prototypes and enterprise-grade reliability, this review outlines the ES-GraphRAG reference architecture. This framework fundamentally reforms graph construction, treating it not as a static snapshot, but as a continuous, materialized projection of an immutable operational event log. By ingesting telemetry through an event bus and appending structural changes with definitive timestamps, ES-GraphRAG enables real "time-travel" querying. During an anomaly, the retrieval layer requests a snapshot of the topology exactly as it existed at the precise incident time, reducing the topology drift gap. Furthermore, ES-GraphRAG replaces dangerous unbounded exploration with a budget-aware retrieval operator that prunes low-value traversals before cost budgets are exceeded, aiming to support more predictable latency through explicit budget enforcement.

The maturation of Generative AIOps demands a disciplined overhaul of system validation. Future work should de-emphasize fluency-biased offline metrics when the target task is causal diagnosis and instead prioritize operationally aligned evaluation protocols. Existing automated RAG evaluation frameworks—such as RAGAs, which provides reference-free metrics for faithfulness, answer relevance, and context relevance [55], and ARES, which fine-tunes lightweight LM judges via prediction-powered inference [56]—offer methodological building blocks that can complement, but cannot substitute for, AIOps-specific operational contracts such as topology-consistency at incident time, snapshot freshness, and bounded retrieval latency. GraphRAG architectures should be engineered and evaluated against rigid, system-level operational contracts. By standardizing the measurement of MTTR reduction, continuous graph construction costs, and strictly bounded retrieval latencies, the field can elevate GraphRAG from a fragile offline retrieval task to a robust, deployment-ready engine for autonomous IT operations.

Acknowledgement: During the preparation of this manuscript, the authors utilized Elicit for prioritization and structured extraction; all inclusion decisions and extracted fields were verified by the authors. Grammarly (v1.0.53.1193) and ChatGPT (GPT 5.2) were used for language editing (grammar, spelling, clarity). All scientific content, analyses, and interpretations are solely those of the authors. The authors have carefully reviewed and revised the output and accept full responsibility for all content.

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

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Ferenc Erdős and Vijayakumar Varadarajan; methodology, Ferenc Erdős and Viorel-Costin Banţa; validation, Viorel-Costin Banţa and Stephen Afrifa; formal analysis, Ferenc Erdős and Viorel-Costin Banţa; writing—original draft preparation, Ferenc Erdős Viorel-Costin Banţa and Vijayakumar Varadarajan; writing—review and editing, Viorel-Costin Banţa and Stephen Afrifa; visualization, Ferenc Erdős supervision, Vijayakumar Varadarajan. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: All reviewed papers are publicly available and cited, primarily sourced from major academic repositories and publishers, including IEEE Xplore, ACM Digital Library, ScienceDirect (Elsevier), SpringerLink, and the arXiv repository.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

Supplementary Materials: The supplementary material is available online at https://www.techscience.com/doi/10.32604/cmc.2026.081005/s1.

Abbreviations

Abbreviation	Full Form
ACL	Access Control List
AI	Artificial Intelligence
AIOps	Artificial Intelligence for IT Operations
CIDs	Canonical Identifiers
CMDB	Configuration Management Database
GNN	Graph Neural Network
GraphRAG	Graph-based Retrieval-Augmented Generation
ES-GraphRAG	Event-Sourced Streaming GraphRAG
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology

ITOps	IT Operations
ITSM	IT Service Management
KG	Knowledge Graph
KPI	Key Performance Indicator
LLM	Large Language Model
MTTR	Mean Time To Repair
RAG	Retrieval Augmented Generation
RCA	Root Cause Analysis
SDG	Service Dependency Graph
SLA	Service-Level Agreement
SLO	Service-Level Objective
SRE	Site Reliability Engineering

Appendix A

Study Selection and Screening. The screening process was conducted on the 139 unique records using Elicit, an AI-powered research assistant. To ensure reproducibility, we utilized a structured two-stage screening protocol:

1. Domain Relevance Screening: We first assessed papers based on their application domain. Papers focused on non-IT domains (e.g., Medical, Legal, Education) were excluded. This step resulted in the removal of 56 papers.
2. Methodological Assessment: The remaining papers were evaluated for the presence of both Knowledge Graph (KG) and Large Language Model (LLM) components. Studies utilizing only standard RAG (without graph structures) or standalone GNNs (without generative capabilities) were excluded. This resulted in the removal of an additional 37 papers.

Results: As shown in Fig. 1 (PRISMA), this process resulted in the exclusion of 93 records. A set of 46 studies met all inclusion criteria and was selected for full-text data extraction.

Table A1: Details of the title/abstract screening process (exclusion/inclusion counts).

Criterion	Definition	Count (Excluded)
IC1 (Inclusion)	**GraphRAG for AIOps:** Paper proposes a system combining Knowledge Graphs and LLMs specifically for IT Operations.	**(Included: 46)**
EC1 (Domain)	**Wrong Domain:** Paper focuses on non-IT Operations domains (other IT fields, or other domains, or general Open-Domain QA).	**56**
EC2 (Method)	**No Graph/LLM Integration:** Paper uses only LLMs (standard RAG) or only Graph Neural Networks (GNNs) without Generative AI integration.	**37**
Total Excluded		**93**

Appendix B

Fields specified for the data extraction.

Table A2: Fields specified for the data extraction.

Scope & positioning	AIOps use case(s)	Incident summarization, RCA, change impact, blast radius, alert correlation, anomaly diagnosis, runbook recommendation, postmortem analysis, other
	Operational setting	Targeted environment: cloud/microservices/Kubernetes, enterprise ITSM, networking, datacenter, etc.); if unclear: "Not reported"
	Contribution type	Method/system/dataset-benchmark/case study/position/unclear
Knowledge-graph construction	Graph type & representation	What graph is used (knowledge graph, service dependency graph, causal graph, incident/event graph)? Representation (property graph vs. RDF vs. other). If not stated, "Not reported".
	Entities & relations (summary)	List of key node types and edge types used (e.g., service–depends_on–service, incident–affects–service, change–deployed_to–service).
	Graph construction method	How is the graph built/updated? (rules/parsing, tracing topology, CMDB/service map ingestion, NLP/IE, LLM extraction, learned edges, etc.). If not stated, "Not reported".
	Temporal & provenance modeling	Does the study model time/provenance? (timestamps, temporal edges, versioning, event-time vs. processing-time, sources tracked). If none, "Not reported".
	Entity resolution/linking across sources	Any service/asset identity resolution, deduplication, ticket-to-service linking, etc.? If none, "Not reported".
GraphRAG retrieval	Retrieval unit	What is retrieved to ground the LLM? nodes/edges, subgraph, paths, communities, summaries, documents linked to graph, hybrid bundle. If unclear, "Not reported".
	Graph retrieval strategy	Describe the main strategy: k-hop neighborhood, constrained traversal, shortest paths, rule-based expansion, random walk, community retrieval/summaries, graph query language, etc.
	Hybrid retrieval & ranking	Any vector search/embeddings, rerankers, graph + vector fusion, GNN embeddings, etc.? If none, "Not reported".

(Continued)

Table A2 (continued)

	Query source & formulation	What initiates retrieval (incident text, alerts, metric symptoms, ticket fields, natural-language question)? Any structured query/planning? If none, "Not reported".
	Bounding for latency/cost	Any constraints for production: bounded hops, top-k, caching, pruning, summarization, streaming updates, cost controls. If none, "Not reported".
Generation patterns	LLM role	summarization, hypothesis/RCA generation, planning/tool use, explanation, report generation, extraction, other
	Output format	Free-form vs. structured template (incident report, RCA hypotheses, blast radius explanation, change impact explanation, runbook steps, etc.). If none, "Not reported".
	Grounding & evidence presentation	How is evidence shown? citations/snippets, referenced nodes/paths, provenance links, quoted logs, "because path X→Y". If none, "Not reported".
Production constraints & governance	Operational constraints beyond latency	Mentioned constraints: streaming/online updates, graph drift, partial observability/missing telemetry, multi-tenancy, access control, PII/governance, reliability. If none, "Not reported".
	Human-in-the-loop/workflow integration	Any analyst loop, UI integration, runbook alignment, ticketing integration, verification steps, feedback learning. If none, "Not reported".
Evaluation & evidence quality	Evaluation setup & baselines	Dataset type (public/private), scale if stated, offline vs. user study, and key baselines (vector RAG, BM25, no-graph, classic AIOps, human). If none, "Not reported".
	Metrics & main results	List of metrics used (retrieval accuracy, groundedness, helpfulness, time-to-triage, etc.) and the main reported outcome (directional is ok). If none, "Not reported".

Appendix C

Study-level coding summary. To ensure transparency and reproducibility, we provide a study-by-study coding sheet (Table A3) that records each of the 31 included studies' assignments under the four-axis framework: graph artifact type (KG vs. SDG), construction logic (topology-first vs. schema-first), retrieval operator family (traversal/query/embedding/hybrid-fusion), and task theme (multi-label: RCA, summarization, remediation, correlation). Task theme is coded as multi-label because a single pipeline may target multiple operational objectives, while the remaining axes are coded as single-label using the dominant implementation described and evaluated in the paper. Table A2 is the source for all aggregate counts and cross-tabulations reported in Section 3.

Table A3: Study-level coding summary.

Study	Task Theme(s)	Graph Artifact	Construction Logic	Retrieval Operator Family	Operational Setting	Contribution Type	Latency Bounding	Other Constraints
Hu et al. (2024) [20]	RCA	SDG	Schema-first	Embedding	Cloud/ microservices	System/ dataset-benchmark	Yes	No
Chen et al. (2026) [38]	Summarization, Remediation	SDG	Topology-first	Traversal	Microservices/ Kubernetes	System	No	No
Zhang (2024) [32]	RCA, Remediation	SDG	Topology-first	Traversal	Cloud-native services, microservic...	System	Yes	Yes
Jha et al. (2026) [37]	RCA, Summarization	SDG	Topology-first	Traversal	cloud infrastructure and microserv...	Method	Yes	No
Zhang et al. (2025) [42]	RCA	SDG	Topology-first	Traversal	Cloud-native systems with microser...	System	Yes	No
Rana et al. (2025) [40]	RCA, Summarization	KG	Topology-first	Hybrid-fusion	Cloud/ microservices/ enterprise ITSM	System	Yes	No
Zheng et al. (2024) [31]	RCA	SDG	Topology-first	Unclear/ not reported	Cloud/ microservices/ Kubernetes	Method	No	Yes
Tang et al. (2025) [41]	RCA	KG	Topology-first	Unclear/ not reported	Cloud/ microservices/ Kubernetes	System	Yes	No
Yu et al. (2026) [46]	RCA, Summarization	SDG	Topology-first	Hybrid-fusion	Cloud systems (Microsoft Azure, Am...	System	Yes	No
Zhang et al. (2026) [33]	RCA, Remediation	SDG	Topology-first	Hybrid-fusion	Cloud-native microservice systems,...	System	Yes	Yes
Liu et al. (2025) [57]	RCA, Remediation, Correlation	KG	Topology-first	Hybrid-fusion	Cloud/microservices	Method	Yes	Yes
Li et al. (2025) [49]	RCA	KG	Topology-first	Unclear/not reported	Networking (BGP routing)	System	No	Yes
Mani et al. (2025) [44]	RCA, Summarization, Remediation, Correlation	SDG	Topology-first	Embedding	Cloud-native microservices archite...	System	Yes	No
Sun et al. (2025) [21]	RCA	KG	Topology-first	Embedding	Optical network management	System	No	Yes
Zhang et al. (2026) [43]	RCA	SDG	Topology-first	Embedding	Microservices/ Kubernetes/ Enterpris...	System	Yes	Yes

(Continued)

Table A3 (continued)

Study	Task Theme(s)	Graph Artifact	Construction Logic	Retrieval Operator Family	Operational Setting	Contribution Type	Latency Bounding	Other Constraints
Ayepah-Mensah et al. (2025) [58]	Unclear/not reported	SDG	Topology-first	Hybrid-fusion	Cloud/ microservices	Method	Yes	Yes
Fu et al. (2026) [59]	RCA, Summarization	KG	Schema-first	Hybrid-fusion	Cloud-native platforms and distrib. . .	System	Yes	No
Cui et al. (2025) [7]	RCA, Summarization	KG	Schema-first	Hybrid-fusion	Cloud (Alibaba Cloud), Networking . . .	Method	No	No
Ren et al. (2026) [60]	RCA, Summarization, Remediation, Correlation	SDG	Topology-first	Hybrid-fusion	networking in smart grids and subs. . .	Method	Yes	Yes
Min et al. (2025) [50]	RCA	KG	Schema-first	Hybrid-fusion	Building operations and maintenanc. . .	System	No	Yes
Mani and Attaranasl (2025) [45]	RCA, Summarization	SDG	Topology-first	Embedding	Cloud-based (AWS)	System	Yes	Yes
Saluja and Bhavsar (2025) [61]	RCA, Summarization	KG	Schema-first	Query	Cloud computing environment, speci. . .	System	No	Yes
Zhang et al. (2025) [35]	RCA	SDG	Topology-first	Unclear/not reported	Microservices	Method	No	No
Zhang et al. (2026) [36]	RCA	SDG	Topology-first	Traversal	Microservices in cloud-native ente. . .	System	Yes	Yes
Xie et al. (2024) [62]	RCA	SDG	Schema-first	Embedding	Cloud systems	System	Yes	No
Zhang et al. (2025) [39]	RCA, Remediation	SDG	Topology-first	Traversal	Cloud/ microservices/ Kubernetes	System/ dataset-benchmark	Yes	Yes
Jiang et al. (2026) [34]	RCA, Summarization, Remediation, Correlation	SDG	Schema-first	Embedding	Industrial systems, particularly f. . .	System	No	No
Pu et al. (2025) [22]	RCA	SDG	Topology-first	Hybrid-fusion	Cloud/ microservices/ Kubernetes, en. . .	System	Yes	No
Xiang et al. (2025) [23]	RCA	SDG	Topology-first	Query	Kubernetes	System	Yes	Yes
Ding et al. (2024) [6]	RCA	SDG	Topology-first	Embedding	Cloud/ microservices/ Kubernetes	Method	Yes	No

(Continued)

Table A3 (continued)

Study	Task Theme(s)	Graph Artifact	Construction Logic	Retrieval Operator Family	Operational Setting	Contribution Type	Latency Bounding	Other Constraints
Gandhi et al. (2025) [63]	Unclear/not reported	SDG	Topology-first	Embedding	Cloud/ microservices	System	Yes	Yes

Note: Task theme is multi-label; other axes are single-label. Some studies use inconsistent terminology (e.g., calling dependency-oriented graphs "knowledge graphs"); coding follows the operational definition in Section 2.3 (topology-centric dependency graphs coded as SDG; multi-entity semantic graphs coded as KG).

References

1. Notaro P, Cardoso J, Gerndt M. A survey of AIOps methods for failure management. ACM Trans Intell Syst Technol. 2021;12(6):1–45. doi:10.1145/3483424.
2. He S, He P, Chen Z, Yang T, Su Y, Lyu MR. A survey on automated log analysis for reliability engineering. ACM Comput Surv. 2022;54(6):1–37. doi:10.1145/3460345.
3. Chen Z, Kang Y, Li L, Zhang X, Zhang H, Xu H, et al. Towards intelligent incident management: why we need it and how we make it. 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. 1487–97. doi:10.1145/3368089.3417055.
4. Soldani J, Brogi A. Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: a survey. ACM Comput Surv. 2023;55(3):1–39. doi:10.1145/3501297.
5. 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.
6. Ding S, Xu Y, Lu Z, Tang F, Li T, Ge J. Power microservices troubleshooting by pretrained language model with multi-source data. In: Proceedings of the 2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA); 2024 Oct 30–Nov 2; Kaifeng, China. p. 1768–75. doi:10.1109/ISPA63168.2024.00241.
7. Cui T, Fu R, Liu C, Ji Y, Gu W, Zhang S, et al. AetherLog: log-based root cause analysis by integrating large language models with knowledge graphs. In: Proceedings of the 2025 IEEE 36th International Symposium on Software Reliability Engineering (ISSRE); 2025 Oct 21–24; São Paulo, Brazil. p. 49–60. doi:10.1109/ISSRE66568.2025.00019.
8. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA. p. 6000–10. doi:10.5555/3295222.3295349.
9. Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems; 2020 Dec 6–12; Vancouver, BC, Canada. p. 9459–74.
10. Gao Y, Xiong Y, Gao X, Jia K, Pan J, Bi Y, et al. Retrieval-augmented generation for large language models: a survey. arXiv:2312.10997. 2003. doi:10.48550/ARXIV.2312.10997.
11. Wu L, Tordsson J, Elmroth E, Kao O. *MicroRCA*: root cause localization of performance issues in microservices. In: NOMS 2020—2020 IEEE/IFIP Network Operations and Management Symposium; 2020 April 20–24; Budapest, Hungary. p. 1–9. doi:10.1109/noms47738.2020.9110353.
12. Huang L, Yu W, Ma W, Zhong W, Feng Z, Wang H, et al. A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions. ACM Trans Inf Syst. 2025;43(2):1–55. doi:10.1145/3703155.
13. Pan S, Luo L, Wang Y, Chen C, Wang J, Wu X. Unifying large language models and knowledge graphs: a roadmap. IEEE Trans Knowl Data Eng. 2024;36(7):3580–99. doi:10.1109/TKDE.2024.3352100.

14. Hogan A, Blomqvist E, Cochez M, D'amato C, De Melo G, Gutierrez C, et al. Knowledge graphs. ACM Comput Surv. 2022;54(4):1–37. doi:10.1145/3447772.
15. Lavrinovics E, Biswas R, Bjerva J, Hose K. Knowledge graphs, large language models, and hallucinations: an NLP perspective. J Web Semant. 2025;85(1):100844. doi:10.1016/j.websem.2024.100844.
16. Peng B, Zhu Y, Liu Y, Bo X, Shi H, Hong C, et al. Graph retrieval-augmented generation: a survey. arXiv:2408.08921. 2024. doi:10.1145/3777378.
17. Edge D, Trinh H, Cheng N, Bradley J, Chao A, Mody A, et al. From local to global: a graph RAG approach to query-focused summarization. arXiv: 2404.16130. 2024.
18. Sun J, Xu C, Tang L, Wang S, Lin C, Gong Y, et al. Think-on-graph: deep and responsible reasoning of large language model on knowledge graph. In: Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024); 2024 May 7–11; Vienna, Austria.
19. Han H, Wang Y, Shomer H, Guo K, Ding J, Lei Y, et al. Retrieval-augmented generation with graphs (GraphRAG). arXiv:2501.00309. 2024.
20. Hu J, Li Y, Xiang Z, Ma L, Jia X, Huang Q. LLM4MDG: leveraging large language model to construct microservices dependency graph. In: Proceedings of the 2024 IEEE 23rd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom); 2024 Dec 17–21; Sanya, China. p. 859–69. doi:10.1109/TrustCom63139.2024.00128.
21. Sun J, Wang H, Zhang B, Li Y, Gao G, Ji Y. OptiAlarmRCAgent: an intelligent agent for alarm root cause analysis in optical networks. In: Proceedings of the 2025 Asia Communications and Photonics Conference (ACP); 2025 Nov 5–8; Suzhou, China. p. 1–5. doi:10.1109/ACP66871.2025.11350409.
22. Pu J, Li Y, Chen Z, Liu J, Jiang Z, Chen J, et al. ErrorPrism: reconstructing error propagation paths in cloud service systems. In: Proceedings of the 2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE); 2025 Nov 16–20; Seoul, Republic of Korea. p. 3534–45. doi:10.1109/ase63991.2025.00292.
23. Xiang Y, Chen CP, Zeng L, Yin W, Liu X, Li H, et al. Simplifying root cause analysis in Kubernetes with StateGraph and LLM. arXiv:2506.02490. 2025.
24. Diaz-de-Arcaya J, Torre-Bastida AI, Zárate G, Miñón R, Almeida A. A joint study of the challenges, opportunities, and roadmap of MLOps and AIOps: a systematic survey. ACM Comput Surv. 2024;56(4):1–30. doi:10.1145/3625289.
25. Yu Q, Zhao N, Li M, Li Z, Wang H, Zhang W, et al. A survey on intelligent management of alerts and incidents in IT services. J Netw Comput Appl. 2024;224(9):103842. doi:10.1016/j.jnca.2024.103842.
26. Page MJ, McKenzie JE, Bossuyt PM, Boutron I, Hoffmann TC, Mulrow CD, et al. The PRISMA, 2020 statement: an updated guideline for reporting systematic reviews. BMJ. 2021;372:n71. doi:10.1136/bmj.n71.
27. Page MJ, Moher D, Bossuyt PM, Boutron I, Hoffmann TC, Mulrow CD, et al. PRISMA, 2020 explanation and elaboration: updated guidance and exemplars for reporting systematic reviews. BMJ. 2021;372:n160. doi:10.1136/bmj.n160.
28. Bernard N, Sagawa Y Jr, Bier N, Lihoreau T, Pazart L, Tannou T. Using artificial intelligence for systematic review: the example of elicit. BMC Med Res Methodol. 2025;25(1):75. doi:10.1186/s12874-025-02528-y.
29. Lau O, Golder S. Comparison of elicit AI and traditional literature searching in evidence syntheses using four case studies. Cochrane Evid Synth Meth. 2025;3(6):e70050. doi:10.1002/cesm.70050.
30. Whitfield S, Hofmann MA. Elicit: AI literature review research assistant. Public Serv Q. 2023;19(3):201–7. doi:10.1080/15228959.2023.2224125.
31. Zheng L, Chen Z, He J, Chen H. MULAN: multi-modal causal structure learning and root cause analysis for microservice systems. In: Proceedings of the ACM Web Conference 2024; 2024 May 13–17; Singapore. p. 4107–16. doi:10.1145/3589334.3645442.
32. Zhang H. A unified AIOps pipeline for joint log-KPI anomaly detection, graph-based root cause localization, and LLM-generated runbooks. J Adv Comput Syst. 2024;4(3):57–73. doi:10.69987/jacs.2024.40305.
33. Zhang W, Yang Z, Peng F, Zhang L, Chen Y, Chen R. GALR: graph-based root cause localization and LLM-assisted recovery for microservice systems. Electronics. 2026;15(1):243. doi:10.3390/electronics15010243.
34. Jiang H, You J, Chen Z, Shi J, Gou H, Ming X, et al. LLM-enhanced intent-aware for proactive decision support services in industrial activities. IEEE Trans Autom Sci Eng. 2026;23:2603–20. doi:10.1109/TASE.2025.3650172.

35. Zhang L, Jia T, Wang K, Hong W, Duan C, He M, et al. Adaptive root cause localization for microservice systems with multi-agent recursion-of-thought. arXiv:2508.20370. 2025.
36. Zhang L, Jia T, Zhai Y, Pan L, Duan C, He M, et al. Hypothesize-then-verify: speculative root cause analysis for microservices with pathwise parallelism. arXiv:2601.02736. 2026.
37. Jha S, Arora R, Bhavya, Zheutlin N, Isaza PT, Shwartz L, et al. Think locally, explain globally: graph-guided LLM investigations via local reasoning and belief propagation. arXiv:2601.17915. 2026.
38. Chen R, Pu Y, Xin J, Wang J, Liao X, Zhang K, et al. GRACE: a strategic LLM-enhanced graph reinforcement learning framework for adaptive fault recovery in microservice systems. In: Aiello M, Deng S, Murillo JM, Georgievski I, Benatallah B, Wang Z, editors. Service-oriented computing. Singapore: Springer Nature; 2026. p. 155–70. doi:10.1007/978-981-95-5012-8_12.
39. Zhang C, Zhang B, Yang D, Peng X, Chen M, Xie S, et al. PromAssistant: leveraging large language models for Text-to-PromQL. arXiv:2503.03114.
40. Rana M, Giri N, Gadde SS. Hyper-automated AIOps for unified operations orchestration in industry. In: 2025 International Conference on Artificial Intelligence for Sustainable Innovation (AI-SI); 2025 Aug 26–28; Kuala Lumpur, Malaysia. p. 1–6. doi:10.1109/AI-SI66213.2025.11340854.
41. Tang L, Kou E, Wang W, Chen Q. A root cause analysis framework for IoT based on dynamic causal graphs assisted by LLMs. IEEE Internet Things J. 2025;12(16):34563–81. doi:10.1109/JIOT.2025.3578512.
42. Zhang X, Wang Q, Li M, Yuan Y, Xiao M, Zhuang F, et al. TAMO: fine-grained root cause analysis via tool-assisted LLM agent with multi-modality observation data in cloud-native systems. IEEE Trans Serv Comput. 2025;18(6):4221–33. doi:10.1109/TSC.2025.3629066.
43. Zhang L, Jia T, Zhai Y, Pan L, Duan C, He M, et al. Agentic memory enhanced recursive reasoning for root cause localization in microservices. arXiv:2601.02732. 2026.
44. Mani N, Attaranasl S, He S. GraphQL-aware healing in service-oriented architectures via multi-signal learning. In: 2025 IEEE International Conference on Service-Oriented System Engineering (SOSE); 2025 Jul 21–24; Tucson, AZ, USA. p. 140–50. doi:10.1109/SOSE67019.2025.00021.
45. Mani N, Attaranasl S. Enhancing adaptive test healing with graph neural networks for dependency-aware decision making. In: 2025 IEEE International Conference on Artificial Intelligence Testing (AITest); 2025 Jul 21–24; Tucson, AZ, USA. p. 126–33. doi:10.1109/AITest66680.2025.00023.
46. Yu G, Mai G, Wang R, Li R, Chen P, Pan L, et al. AlertGuardian: intelligent alert life-cycle management for large-scale cloud systems. arXiv:2601.14912. 2026.
47. Miller RB. Response time in man-computer conversational transactions. In: Proceedings of the December 9–11, 1968, Fall Joint Computer Conference, Part I; 1968 Dec 9–11; San Francisco, CA, USA. p. 267. doi:10.1145/1476589.1476628.
48. Beyer B, Jones C, Murphy NR, Petoff J. Site reliability engineering. Newton, MA, USA: O'Reilly Media, Inc.; 2016.
49. Li P, Liu Y, Su J, Yu B. Understanding Bgp with large language models: towards routing anomaly detection and root cause analysis. 2025 [cited 2026 Jan 26]. Available from: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5278780.
50. Min Z, Bres A, Markov G, Krystallidis A, Neufeld A, Budnik C, et al. Knowledge-graph-centric architecture for reliable fault diagnosis. In: Proceedings of the International Workshop on Middleware for IT/OT Integration; 2025 Dec 15–19; Nashville, TN, USA. p. 37–40. doi:10.1145/3774900.3776640.
51. Akidau T, Begoli E, Chernyak S, Hueske F, Knight K, Knowles K, et al. Watermarks in stream processing systems: semantics and comparative analysis of Apache Flink and Google cloud dataflow. Proc VLDB Endow. 2021;14(12):3135–47. doi:10.14778/3476311.3476389.
52. Hou J, Zhao Z, Wang Z, Lu W, Jin G, Wen D, et al. AeonG: an efficient built-in temporal support in graph databases. Proc VLDB Endow. 2024;17(6):1515–27. doi:10.14778/3648160.3648187.
53. Christophides V, Efthymiou V, Palpanas T, Papadakis G, Stefanidis K. An overview of end-to-end entity resolution for big data. ACM Comput Surv. 2021;53(6):1–42. doi:10.1145/3418896.
54. Bodea AE, Meisenbacher S, Klymenko A, Matthes F. SoK: privacy risks and mitigations in retrieval-augmented generation systems. arXiv:2601.03979. 2026.

55. Es S, James J, Espinosa Anke L, Schockaert S. RAGAs: automated evaluation of retrieval augmented generation. In: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations; 2024 Mar 17–22; St. Julians, Malta. p. 150–8. doi:10.18653/v1/2024.eacl-demo.16.
56. Saad-Falcon J, Khattab O, Potts C, Zaharia M. ARES: an automated evaluation framework for retrieval-augmented generation systems. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers); 2024 Jun 16–21; Mexico City, Mexico. p. 338–54. doi:10.18653/v1/2024.naacl-long.20.
57. Liu L, Ning Y, Sun Z, Liu Y. Multi source alarm root cause localization and self generated work order method based on big language model and dynamic knowledge graph. In: Proceedings of the 2025 IEEE 7th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC); 2025 Dec 5–7; Chongqing, China. p. 948–54. doi:10.1109/IMCEC66174.2025.11332057.
58. Ayepah-Mensah D, Ghebreziabiher AK, Boateng GO, Mizouni R, Mourad A, Otrok H, et al. A RAG-assisted DRL framework for microservices deployment in 6G vehicular networks. In: 2025 21th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob); 2025 Oct 20–22; Marrakesh, Morocco. p. 1–6. doi:10.1109/WiMob66857.2025.11257559.
59. Fu F, Ding H, Qin Y, Yu J, Xu D. Leveraging multi-agent framework for root cause analysis. Complex Intell Syst. 2025;12(1):4. doi:10.1007/s40747-025-02096-0.
60. Ren J, Yao X, Chen H. Transformer substation network disconnection prediction via semantic reasoning with causal modeling. ComSIS. 2026;23(1):321–41. doi:10.2298/csis251027010r.
61. Saluja P, Bhavsar M. KDCFI AI-based knowledge discovery framework for cloud forensic investigation. J Supercomput. 2025;81(17):1572. doi:10.1007/s11227-025-07952-x.
62. Xie Z, Zheng Y, Ottens L, Zhang K, Kozyrakis C, Mace J. Cloud atlas: efficient fault localization for cloud systems using language models and causal insight. arXiv:2407.08694. 2024.
63. Gandhi J, Medicherla RK, Patwardhan M, Sharma D, Naik R. Microservices identification using LLM. In: Proceedings of the 2025 40th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW); 2025 Nov 16–20; Seoul, Republic of Korea. p. 22–5. doi:10.1109/ASEW67777.2025.00013.