The Eastasouth Journal of Information System and Computer Science
Vol. 3, No. 01, August 2025, pp. 151-160
ISSN: 3025-566X, DOI: 10.58812/esiscs.v3i01

Agentic AI-Enhanced Network Performance Monitoring and Diagnostic Analysis for Site Reliability Engineering

Mohit Bajpai
Independent researcher, Alpharetta, GA, USA

Article Info

Article history:

Received Aug, 2025
Revised Aug, 2025
Accepted Aug, 2025

Keywords:

Agentic AI;
AIOps;
Diagnostic Analysis;
Network Performance Monitoring;
Network Reliability
NFRs;
Observability;
Retrieval-Augmented Generation;
Root Cause Analysis;
Site Reliability Engineering;
SLIs;
SLOs;
SRE

ABSTRACT

Network performance monitoring and diagnostic analysis (NPMD) is becoming a core reliability discipline for modern distributed systems because cloud applications, hybrid connectivity, software-defined networking, and multi-region dependency chains can turn small network degradations into visible service incidents. The original paper explained NPMD through Site Reliability Engineering (SRE) concepts such as service level indicators (SLIs), service level objectives (SLOs), and non-functional requirements. This updated version expands the work with a data-driven operating model, stronger references, explicit table and figure captions, and an Agentic AI solution pattern for bounded autonomous diagnosis and remediation. The proposed approach combines telemetry pipelines, SLO evaluation, topology and change evidence, retrieval-augmented diagnostic reasoning, runbook-constrained tool execution, and human approval controls. The paper treats AI as an operational assistant rather than an uncontrolled replacement for SRE judgment: the agent can summarize evidence, correlate probable causes, recommend remediation, and execute only low-risk approved actions while preserving auditability. The result is a practical framework for reducing alert noise, improving time to detect, accelerating incident triage, and strengthening post-incident learning without relying on unsupported claims or unverifiable performance numbers.

Corresponding Author:

Name: Mohit Bajpai
Institution: Independent researcher, Alpharetta, GA, USA
Email: bajpaimohit@gmail.com

1. INTRODUCTION

Digital services now depend on networks that span data centers, public cloud regions, edge locations, third-party APIs, service meshes, content delivery networks, and remote users. In that environment, network reliability is not limited to device uptime. A service can remain technically available while user experience is degraded by high round-trip time, packet retransmission, routing asymmetry, congestion, DNS latency, firewall policy drift, or a misconfigured load balancer. NPMD is therefore a reliability practice that continuously measures service-facing network behavior, diagnoses degradations, and connects operational evidence to corrective action.

SRE provides the governing frame for this practice because it turns reliability into measurable engineering decisions. Google's SRE guidance defines SLOs as reliability targets measured through SLIs and

uses error budgets to balance reliability work with product change velocity [1]-[3]. For NPMD, this means latency, loss, jitter, availability, saturation, and recovery metrics should not be collected only for dashboards; they should be connected to service commitments, alerting policy, incident response, and postmortem learning.

The update in this paper adds an Agentic AI solution layer. Agentic AI refers here to a bounded system that can observe an incident context, retrieve relevant evidence, reason over possible causes, select a next step, call approved diagnostic or remediation tools, and record the outcome. This design is influenced by research on LLM-based autonomous agents, reasoning-and-acting patterns such as ReAct, retrieval-augmented generation, multi-agent collaboration, and feedback-driven agent improvement [4]-[8]. The solution is intentionally constrained by SRE governance, security controls, and human approval gates because autonomous actions in production networks can create outages if they are not scoped and audited.

The remainder of the paper presents a revised NPMD architecture, a data model for network observability, SRE-oriented metrics, Agentic AI diagnostic workflows, governance controls, and implementation guidance. Every table and figure is named and referenced in the surrounding text so that the reader can understand why each artifact appears in the paper.

Figure 1 introduces the reference architecture used throughout the paper. It shows how raw telemetry, synthetic probes, and change evidence flow into an observability store and SLO engine, while a bounded Agentic AI controller uses policy guardrails before recommending or executing remediation.

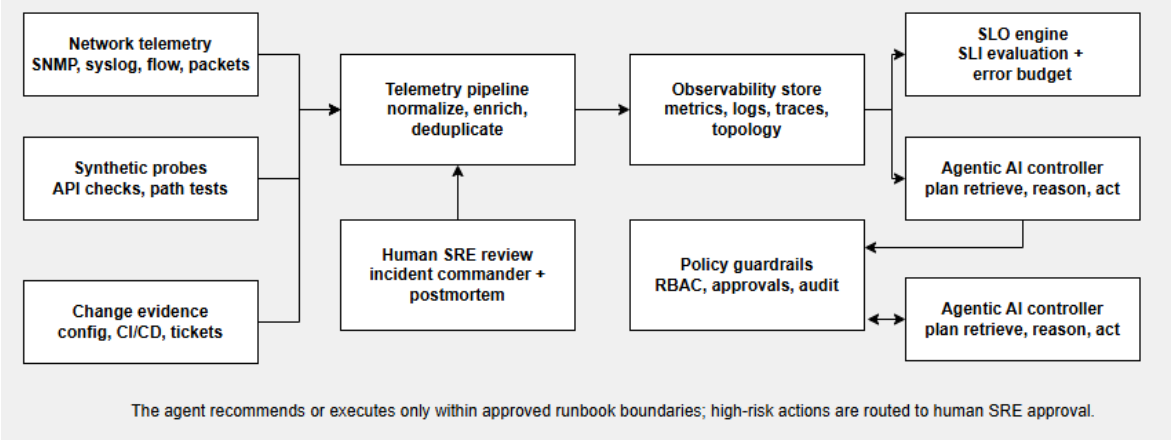

Figure 1. Agentic AI-enhanced NPMD reference architecture for SRE.

2. PROBLEM CONTEXT AND DESIGN REQUIREMENTS

Traditional monitoring often focuses on isolated device counters and threshold alarms. That approach remains useful, but it is insufficient for complex service delivery because users experience end-to-end paths, not individual routers or interfaces. A single incident may involve a configuration change, a cloud interconnect queue, a DNS delay, and a downstream service retry storm. The design goal for modern NPMD is to connect low-level telemetry with service-level impact and operational context.

The architecture must satisfy five requirements. First, it must collect multi-signal telemetry: metrics, logs, traces where available, flow records, synthetic path tests, packet-level evidence, topology metadata, and configuration state. Second, it must normalize data into a consistent schema that can support correlation across devices, applications, sites, and cloud regions. Third, it must evaluate SLOs continuously so that alerts represent user-impacting reliability risk rather than raw noise. Fourth, it must

support diagnosis by preserving the timeline of changes and dependencies around an incident. Fifth, if AI is added, the AI must be bounded by policy, evidence, and approval controls aligned with AI risk management and cybersecurity governance [9]-[11].

Table 1. defines the baseline SRE data model used in the rest of the paper. These values are example targets, not universal guarantees; actual SLOs should be calibrated to business criticality, user location, application type, and contractual requirements.

Table 1. Example SRE-aligned network SLIs, SLOs, and non-functional requirements

Metric	SLI definition	Example SLO target	Operational purpose	Primary evidence
Availability	Percentage of successful network-dependent service checks over a window	>= 99.9% monthly for critical paths	Measures whether users can reach the service	Synthetic probes, load balancer health, service checks
Latency	Round-trip time or request path delay between defined endpoints	p95 < 100 ms for latency-sensitive internal paths; p95 < 300 ms for global paths	Detects degraded user experience before outright failure	Active probes, traces, flow timing, endpoint telemetry
Packet loss	Percentage of packets lost or retransmitted across a measured path	< 0.1% for real-time or critical transactional paths	Identifies congestion, physical errors, or queue drops	Interface counters, active tests, TCP retransmits
Jitter	Variation in packet delay over time	< 30 ms for voice/video-sensitive traffic	Protects timing-sensitive applications	RTP metrics, path probes, endpoint telemetry
Throughput / saturation	Consumed bandwidth as a percentage of usable capacity	Keep sustained utilization below engineered capacity threshold, often 70-80%	Prevents congestion and capacity exhaustion	SNMP/streaming telemetry, NetFlow/IPFIX
Time to detect (TTD)	Elapsed time from first user-impacting symptom to detection	< 5 minutes for tier-1 services	Measures observability effectiveness	Alert timestamps, incident timeline
Time to restore (TTR)	Elapsed time from detection to restoration or mitigation	< 30 minutes for defined critical incidents	Measures response effectiveness	Incident system, runbook output, postmortem
Change correlation	Percentage of incidents with recent change evidence attached	>= 95% of Sev1/Sev2 network incidents	Connects diagnosis to release, configuration, or policy drift	CMDB, Git, ticketing, network config archive

3. DATA ARCHITECTURE FOR NPMD

3.1 Telemetry Collection and Normalization

NPMD begins with instrumentation. Network devices provide interface counters, routing events, tunnel status, hardware health, syslog, traps, flow records, packet captures, and configuration snapshots. Cloud-native systems add service mesh telemetry, API gateway metrics, container networking metrics, load balancer logs, DNS metrics, and distributed traces. OpenTelemetry and similar standards are valuable because they encourage consistent collection and export of observability signals across heterogeneous systems [12]-[14].

Normalization is critical. Device names, interface labels, timestamps, severities, regions, application identifiers, and topology references must

be standardized before correlation. Without normalization, a diagnostic system may treat the same link as multiple assets or fail to connect a service symptom to the device that generated it. Enrichment should add business service ownership, device role, maintenance-window status, software version, cloud region, dependency group, and criticality.

Table 2 names the telemetry sources referenced in the architecture from Figure 1 and explains what each source contributes to diagnostic analysis.

Table 2. NPMD telemetry sources and diagnostic value.

Telemetry source	Examples	Diagnostic value	Retention guidance
Device telemetry	SNMP, gNMI, interface counters, CPU, memory, optics	Identifies saturation, physical errors, device health, and capacity trends	High-resolution short term; summarized long term
Event logs	Syslog, traps, firewall events, AAA logs	Explains state changes, authentication failures, policy hits, and fault messages	Retain raw events for incident window and normalized events for trend analysis
Flow data	NetFlow, IPFIX, VPC flow logs, cloud flow logs	Shows traffic volume, top talkers, denied traffic, and path behavior	Sampled or aggregated based on volume and compliance needs
Synthetic probes	Ping, TCP connect, HTTP checks, DNS checks, path tests	Measures user-like reachability and response from controlled vantage points	Keep granular results around incidents and aggregates for SLOs
Packet evidence	Packet capture, retransmission analysis, TLS handshake timing where permitted	Supports deep diagnosis for loss, fragmentation, handshakes, and protocol errors	Short retention with strict access controls
Topology metadata	CMDB, service map, routing adjacency, cloud resource graph	Connects device symptoms to user-facing services and dependency chains	Continuously updated; versioned for incident reconstruction
Change evidence	Config diffs, deployment tickets, IaC commits, firewall rule changes	Identifies recent operational changes that may explain new symptoms	Version-controlled and linked to incidents
Incident and ticket metadata	Alerts, incident notes, postmortems, escalation history	Provides feedback for future diagnosis and agent retrieval	Retain per compliance and knowledge-management policy

3.2 Processing Pipeline

The processing pipeline should include ingestion, schema mapping, enrichment, deduplication, correlation, storage, and alert generation. Deduplication is important because network incidents often create alarm storms. Correlation groups alerts by time, topology, service dependency, and change proximity. For example, a link flap, OSPF adjacency reset, packet loss spike, and application timeout should be grouped when they affect the same path within the same time window.

The pipeline should preserve both high-cardinality evidence and lower-cardinality SLI outputs. High-cardinality evidence helps engineers investigate specific entities, while SLI outputs drive SLO reporting and error budget policy. Separating these concerns reduces dashboard overload and prevents the SLO layer from becoming a raw telemetry warehouse.

4. DIAGNOSTIC ANALYSIS AND INCIDENT CORRELATION

Diagnostic analysis converts observability data into an operational hypothesis. The recommended workflow begins with symptom confirmation, then service impact assessment, dependency mapping, change review, probable-cause ranking, remediation selection, validation, and postmortem capture. This workflow must be repeatable because incident response quality declines when engineers rely only on memory under pressure.

The diagnostic engine should evaluate events using multiple dimensions: time proximity, topology proximity, service criticality, change proximity, historical recurrence, and blast radius. A packet loss alarm on a low-risk lab interface should not create the same escalation as packet loss on a production interconnect serving payment transactions. Similarly, a configuration commit five minutes before the first symptom should receive higher diagnostic priority than a device warning that has existed for weeks without user impact.

Table 3 provides a named taxonomy that the Agentic AI controller and human SREs can use to classify incidents consistently. This table supports the probable-cause ranking step discussed in this section.

Table 3. Network incident taxonomy for SRE diagnostic workflows

Incident class	Typical signals	Likely causes	Recommended first actions
Path degradation	Higher latency, packet loss, jitter, retransmits	Congestion, routing asymmetry, WAN impairment, cloud interconnect issue	Compare vantage points, inspect flow volume, check recent routing and capacity changes
Configuration drift	New denies, route changes, tunnel failure, asymmetric reachability	Firewall rule change, route policy update, NAT change, IaC deployment	Retrieve config diff, validate intended change, test rollback plan
Capacity exhaustion	Saturation, queue drops, CPU/memory pressure	Traffic growth, attack traffic, batch job, insufficient headroom	Identify top talkers, rate-limit if approved, scale capacity or shift traffic
Control-plane instability	Adjacency resets, route churn, flap events	Protocol instability, device bug, link issue, maintenance activity	Check adjacency history, correlate maintenance, isolate flapping peers
Cloud dependency issue	Region-specific failures, load balancer errors, DNS or API gateway timeouts	Cloud service impairment, misconfigured security group, DNS change	Compare regions, inspect cloud health and resource graph, fail over if SLO risk is high
Security-driven disruption	Denied traffic spike, unusual flows, authentication failures	DDoS, compromised credential, misapplied security control	Engage security runbook, preserve packet/flow evidence, apply approved containment
Monitoring false positive	Alert without synthetic or user-impact confirmation	Bad threshold, broken probe, missing maintenance suppression	Validate with independent evidence, tune alert or suppress during approved window

5. AGENTIC AI SOLUTION FOR NPMD

Agentic AI can add value to NPMD when it is used as an evidence-based diagnostic assistant. The agent should not invent causes, override production policy, or execute unrestricted commands. Instead, it should operate through a controlled loop: observe the incident context, retrieve verified sources, reason over candidate causes, decide whether an approved action exists, act through a limited tool interface, and learn from validated outcomes. Figure 2 names this bounded incident loop.

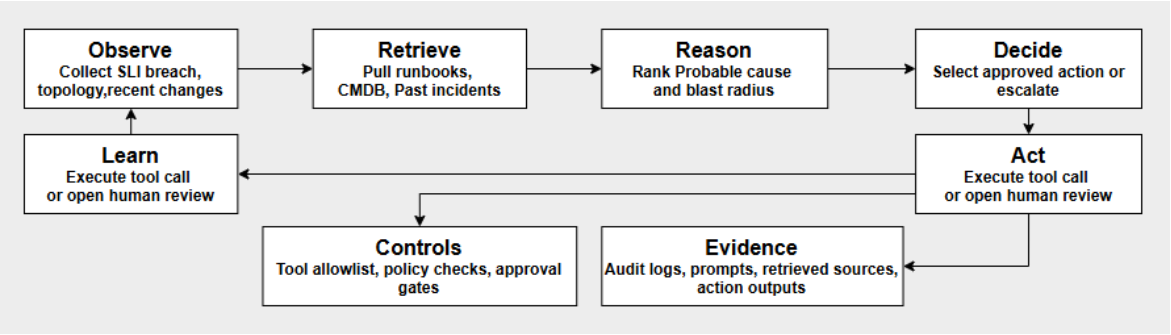

Figure 2. Bounded Agentic AI incident loop for network SRE

The agent architecture should include retrieval-augmented generation so that answers are grounded in runbooks, topology, configuration history, service ownership records, previous incidents, and current telemetry rather than only in model memory. Retrieval-augmented generation was introduced to combine parametric model capability with non-parametric retrieval over external knowledge sources [4]. For incident response, this means the agent should cite the runbook, ticket, configuration diff, or telemetry query that supports each recommendation.

The reasoning-and-action pattern is also important. ReAct demonstrated the value of interleaving reasoning traces with tool actions, allowing a model to gather new observations while solving a task [5]. In NPMD, an equivalent pattern might be: inspect SLO breach, query recent configuration changes, compare affected regions, retrieve the approved runbook, propose mitigation, and then request approval for risky actions. Multi-agent orchestration can separate roles, such as a telemetry analyst agent, change-correlation agent, security reviewer agent, and remediation planner agent, but those agents should share a common policy layer and audit trail [8].

The solution should include feedback without unsafe self-modification. Reflexion-style memory can be useful when it stores validated lessons from postmortems, failed hypotheses, and successful runbook paths [6]. However, production memory should be curated and governed. The system should not store sensitive payloads, secrets, or unreviewed model guesses as durable operational knowledge.

Table 4 names the Agentic AI functions proposed in this paper and separates safe automation from actions that should require human approval.

Table 4. Agentic AI capability matrix for NPMD

Capability	Agent function	Inputs used	Output	Automation boundary
Incident summarization	Condense alert storm into a single timeline	Alerts, SLO breach, topology, incident notes	Human-readable incident brief	Safe to automate with source links

Capability	Agent function	Inputs used	Output	Automation boundary
Evidence retrieval	Find relevant runbooks, prior incidents, config diffs, and telemetry queries	Knowledge base, CMDB, Git, ticketing, observability store	Grounded evidence pack	Safe to automate; must show provenance
Probable-cause ranking	Score hypotheses by topology, time, change, and historical similarity	Telemetry, change history, taxonomy in Table 3	Ranked hypotheses with confidence rationale	Safe as recommendation only
Query generation	Generate read-only telemetry or SQL queries for validation	Schema catalog, approved query templates	Query draft or executed read-only query	Read-only automation allowed after validation rules
Runbook selection	Map incident class to approved runbook	Runbook library, SLO class, service tier	Recommended runbook path	Safe as recommendation
Low-risk remediation	Restart probe, clear stale monitor, suppress known maintenance alert	Approved runbook, maintenance window, RBAC	Executed low-risk action with audit log	Allowed only for pre-approved reversible actions
High-risk remediation	Reroute traffic, roll back config, change firewall policy, rate-limit production traffic	Runbook, approval workflow, blast-radius estimate	Change request or approval prompt	Human approval required
Post-incident learning	Draft postmortem timeline and update knowledge suggestions	Incident transcript, telemetry, approved root cause	Postmortem draft and knowledge-base suggestion	Human review required before durable update

6. GOVERNANCE, SECURITY, AND RELIABILITY CONTROLS

Agentic AI introduces new risks because an agent can combine language reasoning with tool access. The most important control is bounded agency: the agent must have only the permissions required for its assigned task, and high-impact actions must require human approval. OWASP's 2025 guidance for LLM applications includes risks such as prompt injection, sensitive information disclosure, supply chain risk, improper output handling, excessive agency, and unbounded consumption [11]. NIST AI RMF 1.0 also emphasizes governance, mapping, measuring, and managing AI risk across the system lifecycle [9].

For NPMD, prompt injection is especially relevant because operational data can contain attacker-controlled strings, log messages, ticket comments, or configuration descriptions. The agent should treat retrieved text as evidence, not as instructions. Tool calls should be generated through structured schemas, validated against policy, and logged. Secrets must not be included in prompts or model-visible context. Production commands should use just-in-time credentials, command allowlists, and approvals.

Table 5 names the security and reliability controls that should be implemented before an Agentic AI controller is connected to production diagnostic or remediation tools.

Table 5. Governance controls for Agentic AI-enabled NPMD.

Control area	Required control	Why it matters	Evidence to retain
Identity and access	Least-privilege service identity and role-based tool access	Prevents the agent from acting outside approved scope	RBAC policy, access reviews, tool permissions
Tool safety	Allowlisted tools, parameter validation, dry-run mode for change actions	Reduces risk of malformed or unsafe commands	Tool schema, validation logs, dry-run output
Human approval	Mandatory approval for high-risk production actions	Maintains accountability for irreversible or broad-impact changes	Approval record, approver identity, action summary
Prompt and context protection	Separate system instructions from retrieved evidence; sanitize untrusted text	Mitigates prompt injection and instruction override attempts	Prompt template version, retrieved-source metadata
Data protection	Mask secrets, tokens, PII, and packet payloads where not required	Prevents leakage of sensitive operational data	Redaction rules, access logs, data-retention policy
Auditability	Log prompts, retrieved sources, decisions, tool calls, and outputs	Enables incident review and regulatory evidence	Audit trail, correlation ID, immutable event log
Model evaluation	Test against known incidents, false positives, adversarial inputs, and unsafe actions	Validates usefulness and safety before production use	Evaluation report, test cases, failure analysis
Rollback readiness	Require rollback plan for change-producing recommendations	Ensures remediation does not worsen impact	Rollback procedure, change ticket, validation checks

7. IMPLEMENTATION METHODOLOGY

A practical implementation should be incremental. The first phase is observability readiness: inventory critical services, define SLOs, map service dependencies, normalize telemetry labels, and connect monitoring data to incident records. The second phase is diagnostic correlation: implement event grouping, change-correlation, topology-aware alerting, and incident taxonomy. The third phase is AI-assisted triage: deploy the agent in read-only mode to summarize incidents, retrieve evidence, draft queries, and recommend runbooks. The fourth phase is controlled automation: permit low-risk reversible actions while keeping production-impacting actions behind approval gates. The fifth phase is continuous improvement through postmortems, test cases, and SLO review.

The implementation should be measured with operational outcomes, but the organization should avoid claiming improvements until it has baseline and post-implementation data. Suggested measurement categories include alert volume reduction, duplicate alert suppression, mean time to detect, mean time to acknowledge, mean time to mitigate, percentage of incidents with linked change evidence, percentage of incidents with validated root cause, and rate of safe automation success. These metrics should be reported by service tier and incident class because aggregate averages can hide critical-service risk.

8. EVALUATION PLAN AND DATA COLLECTION

Because this paper is an architectural and methodology paper, it does not claim experimentally measured performance gains.

Instead, it proposes an evaluation plan that an organization can use to produce defensible data. The evaluation should compare a baseline period against a pilot period for the same service group, similar traffic patterns, and similar incident severity mix. Each incident should include timestamps for first signal, alert creation, acknowledgement, mitigation, restoration, and postmortem completion. The dataset should also include whether the incident had recent change evidence, whether the Agentic AI recommendation was accepted, whether the recommendation was correct, and whether any tool action was executed.

Recommended evaluation questions include: Did SLO-based alerting reduce non-actionable alerts? Did topology and change correlation reduce diagnosis time? Did the agent retrieve the correct runbook and relevant evidence? Did the agent avoid unsupported conclusions? Were high-risk actions properly escalated? Did postmortem knowledge improve the next similar incident? The answers should be based on incident records and reviewer scoring rather than anecdotal impressions.

9. LIMITATIONS

NPMD cannot remove all uncertainty from distributed systems. Network evidence can be incomplete, sampling may miss short-lived events, packet captures may be restricted by privacy policy, and cloud provider telemetry may not expose all internal states. AI-based diagnosis also has limitations: LLMs can produce plausible but unsupported explanations, retrieved knowledge can be outdated, and agents can take unsafe actions if policy boundaries are weak. For that reason, this paper recommends retrieval grounding, source citation, read-only first deployment, policy-based tool access, human approval for high-risk actions, and continuous evaluation.

Another limitation is that example SLO thresholds in Table 1 are not universal. A trading platform, hospital network, streaming service, internal HR system, and public-sector payment portal may have different tolerances for latency, loss, availability, and recovery time. SLOs must be negotiated with product owners, security teams, network engineering, compliance stakeholders, and customer-facing teams.

10. CONCLUSION

Network Performance Monitoring and Diagnostic Analysis is a foundational SRE practice for modern digital systems. The updated framework presented in this paper expands traditional monitoring into a service-oriented reliability model that combines telemetry, SLO evaluation, topology awareness, change correlation, incident taxonomy, and governance. Agentic AI can strengthen this model by accelerating evidence retrieval, probable-cause ranking, runbook selection, query generation, and postmortem drafting. However, the agent must remain bounded by approved tools, policy gates, human approval for high-risk actions, and auditable evidence.

The central recommendation is to treat Agentic AI as a controlled reliability co-pilot: useful for speed, consistency, and knowledge retrieval, but not a substitute for SRE accountability. When implemented with SLOs, error budgets, security controls, and careful evaluation, Agentic AI-enhanced NPMD can help organizations detect degradations earlier, diagnose incidents more consistently, and improve operational learning across distributed network environments.

REFERENCES

[1] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Eds., Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media, 2016. Available: https://sre.google/sre-book/table-of-contents/

[2] C. Jones, J. Wilkes, N. R. Murphy, and B. Beyer, Eds., The Site Reliability Workbook. O'Reilly Media, 2018. Available: https://sre.google/workbook/table-of-contents/
[3] S. Thurgood, "Example Error Budget Policy," Google SRE Workbook, 2018. Available: https://sre.google/workbook/error-budget-policy/
[4] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," Advances in Neural Information Processing Systems, vol. 33, 2020. Available: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html
[5] S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," International Conference on Learning Representations, 2023. Available: https://openreview.net/forum?id=WE_vluYUL-X
[6] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," Advances in Neural Information Processing Systems, vol. 36, 2023. Available: https://arxiv.org/abs/2303.11366
[7] L. Wang et al., "A survey on large language model based autonomous agents," Frontiers of Computer Science, vol. 18, Art. no. 186345, 2024, doi: 10.1007/s11704-024-40231-1.
[8] Q. Wu et al., "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," Microsoft Research, 2024. Available: https://www.microsoft.com/en-us/research/publication/autogen-enabling-next-gen-llm-applications-via-multi-agent-conversation-framework/
[9] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, 2023, doi: 10.6028/NIST.AI.100-1.
[10] National Institute of Standards and Technology, "The NIST Cybersecurity Framework (CSF) 2.0," NIST CSWP 29, 2024, doi: 10.6028/NIST.CSWP.29.
[11] OWASP Foundation, "OWASP Top 10 for LLM Applications 2025," 2024. Available: https://owasp.org/www-project-top-10-for-large-language-model-applications/
[12] Cloud Native Computing Foundation, "OpenTelemetry becomes a CNCF incubating project," Aug. 26, 2021. Available: https://www.cncf.io/blog/2021/08/26/opentelemetry-becomes-a-cncf-incubating-project/
[13] Cloud Native Computing Foundation, "OpenTelemetry Project Journey Report," Oct. 20, 2023. Available: https://www.cncf.io/reports/opentelemetry-project-journey-report/
[14] OpenTelemetry, "OpenTelemetry specification," 2024. Available: https://github.com/open-telemetry/opentelemetry-specification
[15] Google SRE, "Service Level Objectives," in Site Reliability Engineering. Available: https://sre.google/sre-book/service-level-objectives/
[16] International Telecommunication Union, "Recommendation ITU-T Y.1541: Network performance objectives for IP-based services," 2011. Available: https://www.itu.int/rec/T-REC-Y.1541/