Bian Que: An Agentic Framework with Flexible Skill Arrangement for Online System Operations

Bochao Liu*, Zhipeng Qian*, Yang Zhao*, Xinyuan Jiang*, Zihan Liang, Yufei Ma, Junpeng Zhuang, Ben Chen†, Shuo Yang, Hongen Wan, Yao Wu, Chenyi Lei, Xiao Liang

Kuaishou Technology
benchen4395@gmail.com

Abstract

Operating and maintaining (O&M) large-scale online engine systems (e.g., search, recommendation and advertising) demands substantial human effort for release monitoring, alert response, and root cause analysis. While LLM-based agents are a natural fit for these tasks, a key deployment bottleneck is orchestration: selecting, for each operational event, the relevant data (metrics, logs, change events) and the applicable operational knowledge (handbook rules and practitioner experience). Feeding all signals indiscriminately causes dilution and hallucination, while manually curating the event-to-(data, knowledge) mapping is intractable under dozens of daily releases. We present BIAN QUE, an agentic framework with three contributions: (i) a unified operational paradigm abstracting day-to-day O&M into three canonical patterns: release interception, proactive inspection, and alert root cause analysis; (ii) Flexible Skill Arrangement, in which each Skill specifies the data and knowledge required for a business-module context. Agents generate and update Skills automatically, while on-call engineers can also refine them iteratively through natural-language instructions; (iii) a unified self-evolving mechanism in which a single correction signal drives two parallel pathways: distilling case memory into knowledge, and refining the relevant Skills. Deployed on the e-commerce search engine of a major short-video platform in China, BIAN QUE reduces human-facing alert volume by 75%, achieves 80% root-cause analysis accuracy, cuts mean time to resolution by over 50%, and attains 99.0% cumulative accuracy on 104 offline cases after autonomous initialization and up to three rounds of SRE-assisted correction on the remaining failures. Codes are available at `https://github.com/benchen4395/BianQue_Assistant`.

1 Introduction

Large-scale online engine systems, including search, recommendation, and advertising platforms, form the backbone of modern internet services. Operating and maintaining these systems demands significant engineering effort. At the same time, the underlying modules and services evolve rapidly, with dozens of releases per day. Keeping up with this pace requires deep familiarity with system internals as well as constant attention to runtime behavior, both of which demand significant human effort. Fortunately, the rapid advancement of LLMs and autonomous agents (Openclaw [1], Claude Code [2], Harness engineering [3]) opens new possibilities for automating such operational tasks, offering a scalable path to reducing the manual effort and operational costs that these systems demand.

To apply LLM-based agents in a principled way, we first abstract the heterogeneous operational landscape into a structured set of recurring task types. Despite the apparent diversity of day-to-day

*These authors contributed equally. †The corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

work across business lines and practitioners, the defensive actions engineers actually take follow consistent patterns. We identify three such patterns, which we organize as three lines of defense: monitoring system behavior during releases, proactively inspecting system health on a regular basis, and diagnosing root causes after alerts are triggered. Prior LLM-based O&M agents [4] focus predominantly on the last line—post-alert diagnosis—treating alerting as the starting point. In contrast, our framework places automated release monitoring and proactive inspection ahead of alerting, so that alert-driven diagnosis serves as the last resort rather than the first response.

Underlying all three lines of defense is a consistent cognitive structure: practitioners perceive environmental signals, apply their operational knowledge of system behavior, and reason toward a diagnosis or decision. This renders LLM-based agentic architectures a natural choice. In practice, however, operational scenarios are highly complex and heterogeneous. We refer to the runtime signals emitted by the system as data, including time-series metrics, structured logs, change events, traces, and business indicators. We refer to the distilled operational experience that guides diagnosis as knowledge, including handbook rules, known failure patterns, and module-specific behavioral norms. Feeding all available data wholesale into an LLM makes it difficult for the model to identify what is truly relevant, while knowledge is largely undocumented, shifts with every system iteration, and may draw on entirely different rules depending on the business line, affected module, and operational context. Therefore, a core challenge lies in how to select the right data and knowledge for each operational event, and place them in context. Meanwhile, it is important to ensure that these data and knowledge can be updated or revised after each event is completed, so as to facilitate the rapid handling of similar events in the future. The combinatorial complexity between business lines and scenarios makes it impossible to manually organize such mappings, as it would be an endless and arduous labor.

The above observations motivate three design insights. First, having abstracted the O&M workload into three lines of defence, it is natural to instantiate one specialized agent per line, with each agent dedicated to release interception, proactive inspection, or alert root cause analysis respectively. Second, as systems iterate rapidly and operational contexts shift continuously, the event-to-(data, knowledge) mapping must be kept up to date; maintaining it through natural-language interfaces rather than static configuration makes this process both interpretable and sustainable. Third, because practitioner feedback naturally encodes whether the assembled context was appropriate and what experiential knowledge is worth retaining, a single feedback signal can drive both the mapping and the knowledge to evolve in parallel, eliminating the need for two disconnected update pipelines.

These three insights give rise to BIAN QUE[1], an agentic O&M framework for online engine systems (Figure 1). At its foundation, a unified operational paradigm organizes the three lines of defense under a shared formulation with one specialized agent each; all agents share the same LLM and differ only in their scenario-level knowledge and the Skills they invoke. Built on top of this, Flexible Skill Arrangement lets agents automatically generate and incrementally refine the event-to-(data, knowledge) mapping, with on-call engineers able to intervene through natural-language instructions. Closing the loop, a unified self-evolving mechanism routes each feedback signal into two parallel pathways, memory-to-knowledge distillation and targeted Skill refinement, so that the knowledge base and the mapping co-evolve through a single loop. The contribution lies in the system-level integration of the Skill lifecycle and unified feedback loop; the framework does not fine-tune the underlying LLM. We summarize our contributions as follows:

- We propose a unified operational paradigm that abstracts heterogeneous O&M work into three canonical patterns: release interception, proactive inspection, and alert root-cause analysis.
- We design Flexible Skill Arrangement, where each Skill specifies the relevant data and knowledge for a given business-module context and is generated, updated, and corrected by LLMs through natural-language feedback.
- We introduce a unified self-evolving mechanism in which one feedback signal simultaneously drives memory-to-knowledge distillation and Skill update, enabling the knowledge base and the context-assembly mapping to co-evolve.
- We evaluate BIAN QUE both online and offline. In online deployment on the e-commerce search engine of a short-video platform serving hundreds of millions of users, BIAN QUE

[1]Named after the legendary physician Bian Que in ancient Chinese history, renowned for diagnosing and preventing ailments before symptoms became apparent, mirroring the proactive philosophy of our framework.

reduces human-facing alert volume by 75%, achieves 80% root-cause analysis accuracy, and cuts mean time to resolution by over 50%. In offline evaluation on a curated benchmark of 104 cases, it attains 99.0% cumulative accuracy after autonomous initialization and up to three rounds of SRE-assisted correction.

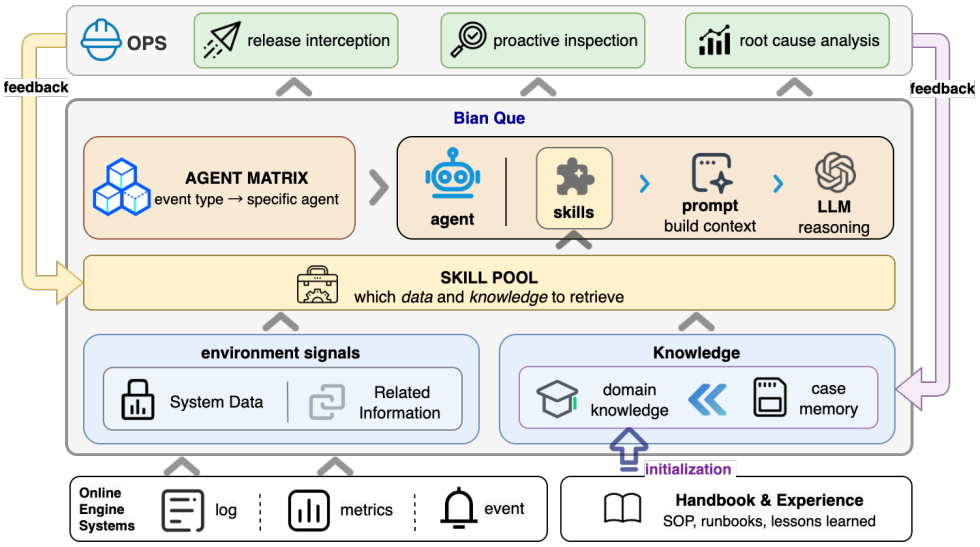


Figure 1: Overview of the BIAN QUE architecture. Operational events from the OPS platform (top) are dispatched to a matching Agent, which invokes one or more matched Skills to assemble the relevant data (system signals: logs, metrics, change events) and knowledge (domain knowledge distilled from case memory, seeded by operational handbooks) for the LLM to reason over; the resulting diagnosis is returned to the OPS platform. Practitioner feedback flows back along two parallel pathways (yellow: Skill refinement; purple: memory-to-knowledge distillation).

2 Related Work

Intelligent Operations and AIOps. Artificial Intelligence for IT Operations (AIOps) [5] is essential for large-scale online engine systems, empowering search, recommendation, and advertising, ensuring the normal operation of the system and algorithms. Early AIOps research [6, 7, 8, 9] addressed individual sub-problems such as anomaly detection [10], log parsing [11], failure prediction [12], and root cause localization [13] in isolation. More recently, LLM-based approaches have tackled log analysis [14], incident summarization [15], conversational troubleshooting [16, 17], and agent-based diagnosis [4, 18], validating the feasibility of applying LLMs to operational scenarios. However, these works share a common positioning: they assume a well-curated input context and focus on reasoning over it, rather than on assembling that context from raw production signals. BIAN QUE is positioned one step earlier in the pipeline, responsible for selecting the right data and knowledge before any diagnostic reasoning happens. A specialized sub-task reasoner like RCAgent [4] can, in principle, be plugged in as the f_{LLM} component of our paradigm (Eq. 1), consuming the $\mathcal{D}(e)$ and $\mathcal{K}(e)$ that our Flexible Skill layer prepares. Our distinctive contributions are accordingly (i) coverage of the operational lifecycle under one unified paradigm, and (ii) treating the (data, knowledge) updating itself, based on both the reasoning and the existing knowledge, as an LLM-generated, continuously evolving artifact.

LLM-based Agents with Evolving Knowledge. LLM-based autonomous agents have established core paradigms for tool use and multi-step reasoning [19, 20, 21, 22, 23, 24, 25]. More recent systems such as LangChain [26], Claude Code [2] and OpenClaw [1] further demonstrate how agentic architectures can leverage standardized tool protocols [27] to operate over large-scale heterogeneous tool ecosystems. These frameworks generally assume that the relevant tools and data sources are either pre-specified or discoverable through exploration, an assumption that breaks down in industrial operations where heterogeneous data sources number in the thousands and the right

subset varies drastically across business lines, modules, and scenarios [28, 29]. Our Flexible Skill mechanism addresses this by introducing an evolvable abstraction layer in which the data-routing logic itself is LLM-generated and incrementally updated through natural-language interfaces. On the knowledge side, prior work spans RAG [30] and its variants [31, 32], memory-augmented architectures [33, 34, 35, 36], and self-refinement methods [37, 38]. Recent work has further explored self-evolving memory systems that learn to update their own memory operations from feedback [39, 40, 41]. However, conventional RAG treats the knowledge base as static, and most self-evolving mechanisms operate along a single feedback pathway—improving either the knowledge store or the agent behavior, but not both. Our framework introduces a unified feedback mechanism in which each correction signal simultaneously drives memory-to-knowledge distillation and Skill refinement, enabling the two to co-evolve through a shared loop.

The claimed contribution is the system-level protocol for maintaining event-specific context through the Skill lifecycle and unified feedback, rather than a new indexing method or LLM architecture.

3 Method

3.1 Problem Formulation and Architectural Overview

The operational maintenance of large-scale online engine systems (search, recommendation, advertising) gives rise to a stream of operational events e, including release deployments, alert triggers, and scheduled inspection requests. For each event, the agent produces a structured output o that includes a decision (e.g., "rollback" or "safe to proceed"), supporting evidence, and recommended follow-up actions. As explained in §1, we identify three canonical patterns—release interception, proactive inspection, and alert root cause analysis—that collectively cover the bulk of the operational workload. All three share a common computational structure:

$$o = f_{\mathrm{LLM}}\big(\mathcal{D}(e), \mathcal{K}(e), p_s\big), \tag{1}$$

where $\mathcal{D}(e)$ denotes the data retrieved for event e, $\mathcal{K}(e)$ denotes the applicable operational knowledge, p_s denotes the scenario-specific prompt composition, and f_{LLM} the large language model performing structured reasoning. The prompt p_s is further decomposed into an agent-level and one or more Skill-level components, which are depicted in Eq. 2.

Figure 1 maps Eq. 1 onto the architecture: existing engine infrastructure supplies $\mathcal{D}$, the two-phase knowledge store (§2.4) supplies $\mathcal{K}$, and a general-purpose LLM serves as f_{LLM}. We reuse the industrial data stack; selecting the relevant signals is delegated to the Flexible Skill mechanism (§2.3). The LLM is a pluggable reasoning module, not a component to be fine-tuned. Appendix A supports higher-capacity backbones on the evaluated tasks, without establishing a universal parameter threshold.

3.2 Agent Matrix and Skill Pool

A monolithic agent cannot efficiently handle the full breadth of online engine maintenance, because the data requirements and knowledge contexts vary drastically across business lines and service modules. We therefore adopt a matrix design that decomposes the problem along two orthogonal dimensions (shown in Figure 2).

Agents as Scenario Abstractions. Each agent type corresponds to one canonical pattern: the release interception agent implements checkpoint-based evaluation, the inspection agent implements periodic comprehensive analysis, and the alert agent implements event-triggered diagnosis.

Skills as Business-Module Orchestration Units. Each Skill specifies the data and knowledge needed for a particular combination of business and module. For instance, a Skill for "recommendation recall module availability" declares which metrics, logs, and change events to retrieve, which knowledge-base entries to consult (e.g., "this module's GMV impact is typically small"), and how to reason over both. The internal structure of Skills is detailed in §2.3.

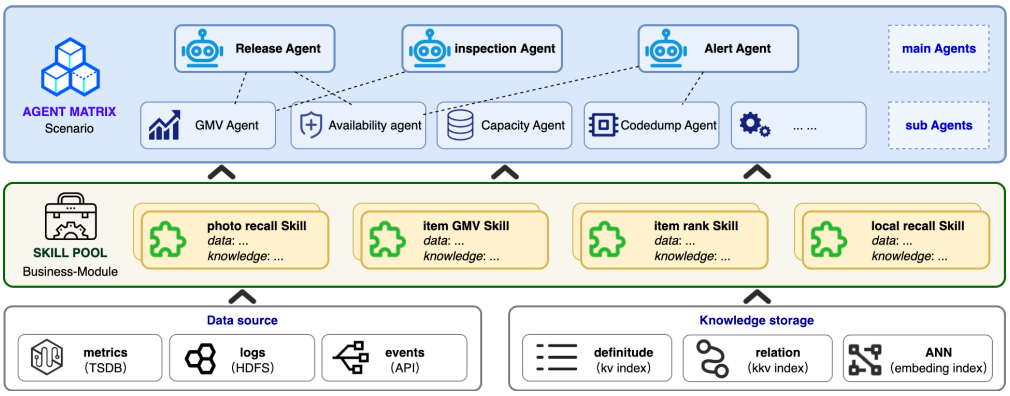


Figure 2: Agent Matrix and Skill Pool. Each Agent (top) implements one canonical pattern; Skills (middle) are business-module orchestration units in a shared pool. A single execution composes one Agent with matched Skills, which call into the knowledge store and data sources (bottom).

Orthogonal Composition. A single agent execution composes the agent's scenario logic with one or more matched Skills:

$$o = f_{\text{LLM}}\left(\bigcup_i \mathcal{D}_{s_i}(e), \bigcup_i \mathcal{K}_{s_i}(e), p_{\text{agent}}, \{p_{s_i}\}\right), \quad \{s_i\} = \text{MATCH}(e, \mathcal{S}), \tag{2}$$

where $\mathcal{S}$ is the Skill pool. MATCH performs exact lookup on a key parsed from event type/module metadata, using the same rule for storage and query; an unseen key triggers generation. The scenario prompt in Eq. 1 composes p_{agent} with $\{p_{s_i}\}$. Agents can improve scenario reasoning independently of Skills, while Skills can evolve their data and knowledge configurations independently of agent logic. Inspection and alert sub-agents may further specialize p_{agent} by metric family (GMV, capacity, availability, coredump); they share the parent's scenario logic and are omitted from Eq. 2.

3.3 Flexible Skill Arrangement

The Skill abstraction provides precise, scenario-appropriate context. Raw data can dilute critical signals, while overly restrictive configurations miss information. Combinatorial business/module/scenario diversity and rapid system evolution make manual curation unsustainable. Flexible Skills are therefore LLM-generated, LLM-updated, and human-correctable.

3.3.1 Skill Structure

Each Skill is a structured document with three fields:

$$\text{Skill} = \langle \text{LoadDataSchema}, \text{Prompt}, \text{Meta} \rangle. \tag{3}$$

LoadDataSchema is a JSON specification that declares what to retrieve. It lists the data-source calls (e.g., time-series metrics, structured logs, change events, traces) and knowledge-base queries to execute, along with their parameters and whether each source is mandatory or optional. Prompt is a structured template that defines how to reason. It guides the LLM through a multi-step analysis over the retrieved data and knowledge to produce a structured diagnosis. At execution time, the Skill-level Prompt composes with the agent-level p_{agent}, and p_{agent} supplies scenario logic (e.g., how to structure a release verdict), while the Skill Prompt supplies business-module-specific analysis steps. Meta contains name, version, description, and tags used for metadata-based Skill matching (§2.2).

3.3.2 Skill Lifecycle

The lifecycle consists of two phases (Figure 3): Generation creates a new Skill when no matching one exists for a given business-module combination, and Update incrementally revises an existing Skill in response to practitioner feedback.

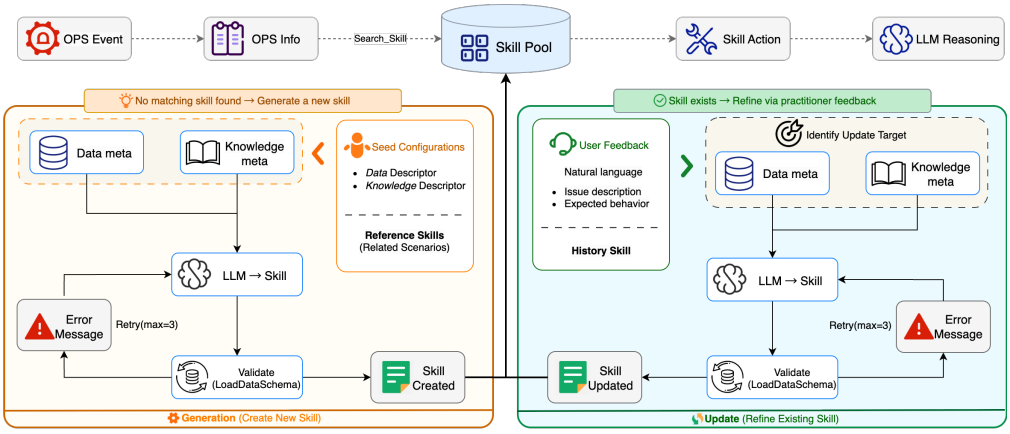


Figure 3: Flexible Skill lifecycle. New Skills are generated from seed configurations with validation and retry. Existing Skills are updated via practitioner feedback through a natural-language interface. Executable schema changes are validated; prompt-only changes have no automatic semantic-correctness gate.

Generation. A first event with no matching business-module Skill triggers generation. The LLM receives a source descriptor listing available data sources and knowledge entries with their semantics, and a capability descriptor specifying scenario-relevant sources and their relationships. The generated LoadDataSchema is executed against the triggering event; inaccessible sources or missing required fields are returned to the LLM for regeneration, up to three retries. Existing Skills from related scenarios provide references for cross-scenario transfer.

Update. Once deployed, a Skill may need to evolve as systems change. When a practitioner identifies a gap or an error in the agent's analysis, they describe the issue and expected behavior in natural language through a messaging interface. The framework then revises the appropriate component of the Skill—adjusting LoadDataSchema if the agent missed a critical data source or knowledge entry, or adjusting the Prompt if the agent retrieved adequate information but reasoned over it incorrectly. Schema changes follow the validate-and-retry loop to check source accessibility and required fields, not diagnostic correctness. Prompt-only updates have no automatic semantic-correctness gate; their effects are monitored on subsequent cases and reviewed by senior SREs. These changes require no code edits.

3.4 Self-Evolving Knowledge and Skill via Unified Feedback

A central design choice of BIAN QUE is that one feedback signal simultaneously drives two parallel learning pathways: one evolving the knowledge base, one evolving the Skills. Every agent execution produces a case record $c = (e, \{s_i\}, \mathcal{D}, \mathcal{K}, o, f)$, where f is the practitioner's feedback when available. The feedback is dispatched along two pathways:

$$f \longrightarrow \begin{cases} \text{MemoryWrite}(c) \rightarrow \text{KnowledgeDistill}(c), \\ \text{SkillRefine}(\{s_i\}, f). \end{cases} \tag{4}$$

The knowledge pathway governs the accumulation of distilled operational experience, such as failure patterns and module-specific behavioral norms; the Skill pathway governs which data and knowledge the agent retrieves and how it reasons over them for each event.

3.4.1 Knowledge Pathway

The knowledge pathway has two phases. Each execution is first written to a short-term case-memory store; relevant recent cases are retrieved before reasoning to recognize recurring failures. Each new case also triggers LLM extraction of reusable causal relationships, failure patterns, and metric-interpretation rules into long-term knowledge. Daily offline consolidation deduplicates, prunes, and resolves contradictions, while corrections take effect immediately through case memory. A small

set of operational-handbook entries seeds the same store at deployment. Three indices support retrieval: KV for facts keyed by business/scenario identifiers; KKV for relational knowledge, such as module-to-module impacts for a metric; and a vector index for semantic retrieval. Skills query these indices through LoadDataSchema (§2.3), coupling retrieval and knowledge at the query-specification level.

After six months, one business line maintains approximately 50 long-term entries, with about five added or revised weekly and observed end-to-end retrieval latency below 100 ms.

3.4.2 Skill Pathway

The same feedback signal identifies the source of an error and selects the appropriate lifecycle operation (§2.3). Missing data or a wrong knowledge query leads to a LoadDataSchema update; flawed analysis of adequate evidence leads to a Prompt update. Incorrect or stale knowledge content instead follows the knowledge pathway, with immediate case-memory correction and later long-term consolidation. Schema edits undergo the execution check; prompt-only edits have no automatic semantic gate. Routing combines predefined failure categories with LLM interpretation, with explicit SRE instructions taking precedence. Historical versions support manual restoration; automatic rollback is not implemented.

3.4.3 Co-Evolution of Knowledge and Skills

The two pathways above are coupled by construction: a richer knowledge base provides more reference entries for newly generated or updated Skills, while higher-quality Skills produce cleaner case records for knowledge distillation. We qualitatively observe this compounding effect in deployment; measuring its long-term effect remains future work. A targeted intervention on six residual failures resolves 4/6 cases through knowledge-only updates, 3/6 through Skill-only updates, and 5/6 through joint updates (Appendix E); this is case-level evidence, not a general estimate of compounding gains.

4 Experiments

4.1 Experimental Setup

Implementation Details. BIAN QUE has operated for over six months on the platform's e-commerce search engine, serving hundreds of millions of daily active users. Dozens of service modules are maintained by approximately 100 engineers, with dozens of daily releases. The default backbone is Qwen3.5-35B-FP8 on one NVIDIA Tesla L20 GPU. Skills are YAML documents indexed by metadata keys; Redis provides KV/KKV knowledge indices and an ANN service provides vector retrieval. Case memory uses a rolling window with daily consolidation. All pass@k evaluations use temperature $T = 0.3$; other experiments use greedy decoding unless noted. Each evaluation run resets the Skill directory to the same clean seed state to avoid cross-run leakage.

Datasets and Evaluation Metrics. Table 1 summarizes the three patterns and where each is evaluated. We adopt four metrics. Alert Volume Reduction measures the relative decrease in alerts requiring human attention after deployment. RCA Accuracy is the proportion of root-cause analyses confirmed correct by practitioners. MTTR (Mean Time to Resolution) is the average time from alert trigger to issue resolution. Autonomous pass@k runs each event k times and counts it as passed if at least one run produces a correct Skill yielding the ground-truth diagnosis.

We reserve pass@k for autonomous sampling. In the human-assisted results, correction@r measures residual cases solved within r SRE-correction rounds, and HITL-acc@r measures cumulative accuracy over all 104 cases. The original single-run counts below are retained separately from the three-run means in Appendix C. For offline attention labels, three SREs agreed unanimously on 100/104 cases (96.2%); four were jointly adjudicated. This does not measure agreement on free-text diagnoses.

4.2 Online Production Deployment Results

Table 2 reports the key operational metrics before and after BIAN QUE deployment over a six-month evaluation period. Relative numbers are normalized to the pre-deployment baseline. The third row

Table 1: The three canonical operational patterns and their evaluation in this paper.

Pattern	Target problem	Core capability	Evaluated in
Release interception	Incidents from releases	Single-event detection	Online; App. D
Proactive inspection	Latent system risks	Multi-event analysis	pass@k
Alert root cause analysis	Post-alert diagnosis	Correlation-based attribution	pass@k; RCA; MTTR

(absolute non-actionable alerts) is the compound of rows 1 and 2 and is the most directly interpretable measure of practitioner pager load.

Table 2: Online production deployment over a six-month window, comparing the month preceding rollout with the six-month post-rollout period.

Metric	Pre-deployment	Post-deployment
Human-facing alert volume (relative)	100%	25% (↓75%)
Non-actionable ratio	80%	15%
Non-actionable alerts, absolute (relative)	100%	∼5% (↓∼95%)
RCA accuracy	—	80%
Diagnosis returned within 5 minutes	—	95% of cases
MTTR (relative)	100%	$<$ 50% (↓50%+)

Alert load reduction combines two mechanisms. Release interception and proactive inspection act upstream, while alert diagnosis suppresses human-facing notifications for automatically handled cases. As clarified in the deployment protocol, the underlying source alert remains active. Table 2 retains the reported relative values: the compound non-actionable volume is approximately $0.25 \times 0.15/0.80 \approx 0.047$ of baseline when the ratios refer to the same notification population. This compound quantity describes pager load, not an independently measured source-alert reduction.

Downstream diagnostic quality. For alerts warranting attention, the agent returns a structured diagnosis within five minutes in 95% of cases, with 80% RCA accuracy. Reported MTTR falls by over 50%, alongside faster diagnosis and lower practitioner cognitive load from evidence-backed recommendations. Incorrect diagnoses primarily involve newly deployed services with insufficient domain knowledge, motivating the feedback mechanism (§2.4) and ablations (§3.4).

These six-month results are descriptive before–after observations. The system retains human oversight: P0 alerts are never suppressed; persistent or unhandled cases are escalated. Appendix F details the safeguards, a matched 42-alert replay, and a non-randomized knowledge-transition comparison.

4.3 Flexible Skill Mechanism Evaluation

Skill Initialization. We curate 104 production events—44 alerts and 60 inspections—with ground-truth diagnoses annotated by senior SREs. Table 3 retains the original single-run Skill-generation results: pass@1 is 78.8%, and five autonomous samples reach 94.2%, leaving six failures. The source and capability descriptors scaffold initialization; remaining failures involve ambiguous attribution or business knowledge absent from the Skill pool, motivating human-in-the-loop correction.

Table 3: Skill initialization on 104 real operational events (original single run). pass@k is the fraction solved by at least one of k independently sampled autonomous generations.

Scenario	Total	pass@1	pass@2	pass@3	pass@4	pass@5
Alert	44	31 (70.5%)	36 (81.8%)	40 (90.9%)	42 (95.5%)	42 (95.5%)
Inspection	60	51 (85.0%)	52 (86.7%)	54 (90.0%)	55 (91.7%)	56 (93.3%)
Overall	104	82 (78.8%)	88 (84.6%)	94 (90.4%)	97 (93.3%)	98 (94.2%)

Repeated evaluation and external baselines. Three independent runs give $77.9 \pm 2.5\%$ pass@1 and $94.6 \pm 0.6\%$ pass@5. Appendix C reports uncertainty and comparisons with frozen Skills, Naive RAG, and a Voyager-style O&M adaptation under shared snapshots, model, budget, and evaluator. These repeated results are distinct from the original single-run counts in Table 3.

Human-in-the-Loop Correction. Senior SREs provided targeted natural-language corrections for the six failures in Table 3 (two alerts, four inspections). Table 4 reports correction@r on these cases, while Table 5 reports cumulative HITL-acc@r on all 104 events. One correction round raises cumulative accuracy from 94.2% to 97.1%; three rounds reach 99.0%. Alert cases reach 100% by the third correction; inspection cases plateau at 98.3% cumulative accuracy, with one case still unresolved. These outcomes include human assistance and are distinct from autonomous pass@k.

Table 4: Human-in-the-loop correction on the six failures at autonomous pass@5. Columns count SRE-correction rounds; the original case counts are unchanged.

Scenario	Total	correction@1	correction@2	correction@3	correction@4	correction@5
Alert	2	1 (50.0%)	1 (50.0%)	2 (100.0%)	2 (100.0%)	2 (100.0%)
Inspection	4	2 (50.0%)	3 (75.0%)	3 (75.0%)	3 (75.0%)	3 (75.0%)
Overall	6	3 (50.0%)	4 (66.7%)	5 (83.3%)	5 (83.3%)	5 (83.3%)

Table 5: Cumulative accuracy on 104 events after autonomous initialization (up to five generations), followed by r SRE-correction rounds on the remaining failures.

Scenario	Total	HITL-acc@1	HITL-acc@2	HITL-acc@3	HITL-acc@4	HITL-acc@5
Alert	44	43 (97.7%)	43 (97.7%)	44 (100.0%)	44 (100.0%)	44 (100.0%)
Inspection	60	58 (96.7%)	59 (98.3%)	59 (98.3%)	59 (98.3%)	59 (98.3%)
Overall	104	101 (97.1%)	102 (98.1%)	103 (99.0%)	103 (99.0%)	103 (99.0%)

4.4 Ablation Study

We evaluate two ablated variants on the same 104-event dataset used in §3.3, under the same pass@k protocol: 1) STATIC: Skills are manually authored and fixed, disabling LLM-driven generation and update. This isolates the contribution of the Flexible Skill mechanism (§2.3). 2) NOKNOW: No historical knowledge is retrieved during reasoning, disabling both short-term case-memory retrieval and long-term distilled-knowledge retrieval (§2.4). Tables 6 and 7 report pass@k for each variant. Both can be compared directly against the full-framework results in Table 3.

Flexible Skill is the most critical component. In Table 6, fixed Skills reduce overall pass@5 from 94.2% to 83.7% and pass@1 from 78.8% to 65.3% in the originally reported run. Alert cases show a larger pass@5 decrease than inspection cases (13.7 vs. 8.3 percentage points). Residual failures at pass@5 rise from 5.8% to 16.3%, indicating coverage gaps within this sampling budget.

Table 6: STATIC Skills: manually authored without LLM-driven generation or updates, tested on the 104-event dataset. Original single-run values are retained.

Scenario	Total	pass@1	pass@2	pass@3	pass@4	pass@5
Alert	44	26 (59.1%)	30 (68.2%)	33 (75.0%)	34 (77.3%)	36 (81.8%)
Inspection	60	42 (70.0%)	45 (75.0%)	46 (76.7%)	51 (85.0%)	51 (85.0%)
Overall	104	68 (65.4%)	75 (72.1%)	79 (76.0%)	85 (81.7%)	87 (83.7%)

Historical knowledge is irreplaceable. Table 7 shows pass@5 falling from 94.2% to 86.5% without knowledge retrieval, a 7.7-point decrease, alongside a 7.6-point decrease at pass@1. The gap persists within five samples, supporting the importance of historical incidents and service-specific interpretation rules.

Table 7: NOKNOW: both case-memory and distilled-knowledge retrieval are disabled, tested on the 104-event dataset (original single run).

Scenario	Total	pass@1	pass@2	pass@3	pass@4	pass@5
Alert	44	28 (63.6%)	34 (77.3%)	37 (84.1%)	39 (88.6%)	40 (90.9%)
Inspection	60	46 (76.7%)	46 (76.7%)	48 (80.0%)	49 (81.7%)	50 (83.3%)
Overall	104	74 (71.2%)	80 (76.9%)	85 (81.7%)	88 (84.6%)	90 (86.5%)

Feedback-driven refinement sustains long-term accuracy. Offline, one SRE-correction round raises cumulative accuracy from 94.2% to 97.1%; three reach 99.0%. Online, both arms process the same event stream for 13 days from identical Skill and knowledge snapshots. Only the adaptive arm incorporates feedback. Frozen-arm accuracy declines from approximately 75% to 32%; adaptive accuracy stays around or above 80% for most days, recovering within two days of the day-7 dip. Rotating on-call SREs may both evaluate and correct cases; Appendix F details this limitation and the weekly audits.

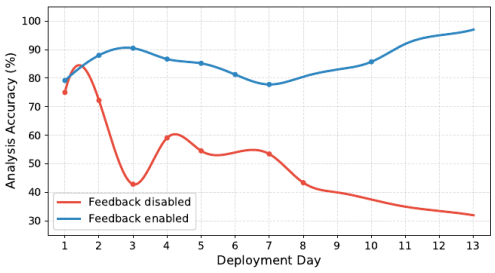


Figure 4: Day-level alert-analysis accuracy on production traffic (original 13-day study). Red: feedback disabled; blue: feedback enabled.

An additional 57-case release-interception benchmark covers the remaining operational pattern (Appendix D). The six-case pathway intervention (Appendix E) and approximately one-month advertising/recommendation deployments (Appendix G) provide targeted complementary evidence.

Broader Impacts

BIAN QUE can reduce repetitive operational work and notification fatigue, shorten incident diagnosis, and improve service reliability. Incorrect diagnoses or excessive reliance on automated analysis could instead delay incident response; routinely delegating reasoning may also erode engineers' operational expertise, while operational logs and incident knowledge can contain confidential information. In the evaluated deployment, the system provides diagnoses without autonomously executing remediation, retains source alerts, escalates persistent or unhandled issues, and never suppresses P0 notifications. Human review and correction remain necessary; these safeguards reduce risk but do not establish that failures are impossible. Proprietary production records are not publicly released.

5 Conclusion

We have presented BIAN QUE, an agentic framework for automated intelligent operations of large-scale online engine systems. The framework makes three contributions: a unified operational paradigm that provides complete yet lightweight coverage of the majority of maintenance tasks through three canonical patterns; a Flexible Skill Arrangement mechanism that jointly encapsulates data routing and domain knowledge and supports LLM-driven Generation and Update; and a unified feedback mechanism that simultaneously drives knowledge base evolution and Skill refinement through practitioner signals. Production deployment on a large-scale search engine demonstrates substantial operational improvements: a 75% reduction in human-facing alert volume, a roughly 95% reduction in practitioner-facing non-actionable alerts, 80% RCA accuracy on the alerts that do require attention, and over 50% MTTR compression.

Acknowledgments and Disclosure of Funding

We thank Kuaishou Technology for supporting this work, which was conducted at Kuaishou Technology.

Funding. This work was supported by Kuaishou Technology. The authors received no other third-party funding or support for this work during the 36 months preceding submission.

Competing interests. The authors' affiliation with Kuaishou Technology is disclosed in the author information. The authors declare no additional financial relationships or competing interests relevant to this work during the 36 months preceding submission.

References

[1] OpenClaw. Openclaw. `https://github.com/openclaw/openclaw`, 2026. Open-source personal AI assistant, version 2026.3.8, accessed 2026-03-09.

[2] Anthropic. Claude code overview. `https://code.claude.com/docs/en/overview`, 2026. Official documentation, accessed 2026-03-10.

[3] OpenAI. Harness engineering: leveraging codex in an agent-first world. Engineering blog, 2026. URL `https://openai.com/index/harness-engineering/`. Published: 2026-02-11. Accessed: 2026-03-13.

[4] Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Jihong Wang, Fengbin Yin, Lunting Fan, Lingfei Wu, and Qingsong Wen. Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. In Proceedings of the 33rd ACM international conference on information and knowledge management, pages 4966–4974, 2024.

[5] Lingzhe Zhang, Tong Jia, Mengxi Jia, Yifan Wu, Aiwei Liu, Yong Yang, Zhonghai Wu, Xuming Hu, Philip S Yu, and Ying Li. A survey of aiops for failure management in the era of large language models. arXiv preprint arXiv:2406.11213, 2024.

[6] Paolo Notaro, Jorge Cardoso, and Michael Gerndt. A survey of aiops methods for failure management. ACM Transactions on Intelligent Systems and Technology (TIST), 12(6):1–45, 2021.

[7] Yinfang Chen, Jiaqi Pan, Jackson Clark, Yiming Su, Noah Zheutlin, Bhavya Bhavya, Rohan Arora, Yu Deng, Saurabh Jha, and Tianyin Xu. Stratus: A multi-agent system for autonomous reliability engineering of modern clouds. arXiv preprint arXiv:2506.02009, 2025.

[8] Evelien Riddell, James Riddell, Gengyi Sun, Micha´L Antkiewicz, and Krzysztof Czarnecki. Stalled, biased, and confused: Uncovering reasoning failures in llms for cloud-based root cause analysis. arXiv preprint arXiv:2601.22208, 2026.

[9] Arthur Vitui and Tse-Hsun Chen. Empowering aiops: Leveraging large language models for it operations management. arXiv preprint arXiv:2501.12461, 2025.

[10] Mohammad Braei and Sebastian Wagner. Anomaly detection in univariate time-series: A survey on the state-of-the-art. arXiv preprint arXiv:2004.00433, 2020.

[11] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. Drain: An online log parsing approach with fixed depth tree. In 2017 IEEE international conference on web services (ICWS), pages 33–40. IEEE, 2017.

[12] Qingwei Lin, Ken Hsieh, Yingnong Dang, Hongyu Zhang, Kaixin Sui, Yong Xu, Jian-Guang Lou, Chenggang Li, Youjiang Wu, Randolph Yao, et al. Predicting node failure in cloud service systems. In Proceedings of the 2018 26th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, pages 480–490, 2018.

[13] Jacopo Soldani and Antonio Brogi. Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey. ACM Computing Surveys (CSUR), 55(3):1–39, 2022.

[14] Siraaj Akhtar, Saad Khan, and Simon Parkinson. Llm-based event log analysis techniques: A survey. arXiv preprint arXiv:2502.00677, 2025.

[15] Paulina Toro Isaza, Michael Nidd, Noah Zheutlin, Jae-wook Ahn, Chidansh Amitkumar Bhatt, Yu Deng, Ruchi Mahindru, Martin Franz, Hans Florian, and Salim Roukos. Retrieval augmented generation-based incident resolution recommendation system for it support. arXiv preprint arXiv:2409.13707, 2024.

[16] Devjeet Roy, Xuchao Zhang, Rashi Bhave, Chetan Bansal, Pedro Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. Exploring llm-based agents for root cause analysis. In Companion proceedings of the 32nd ACM international conference on the foundations of software engineering, pages 208–219, 2024.

[17] Jonathan Pan, Wong Swee Liang, and Yuan Yidi. Raglog: Log anomaly detection using retrieval augmented generation. In 2024 IEEE World Forum on Public Safety Technology (WFPST), pages 169–174. IEEE, 2024.

[18] Lingzhe Zhang, Tong Jia, Mengxi Jia, Yifan Wu, Aiwei Liu, Yong Yang, Zhonghai Wu, Xuming Hu, Philip Yu, and Ying Li. A survey of aiops in the era of large language models. ACM Computing Surveys, 58(2):1–35, 2025.

[19] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In The eleventh international conference on learning representations, 2022.

[20] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. Advances in neural information processing systems, 36: 68539–68551, 2023.

[21] Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and additional opinions. arXiv preprint arXiv:2306.02224, 2023.

[22] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. Advances in Neural Information Processing Systems, 36:38154–38180, 2023.

[23] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. arXiv preprint arXiv:2305.16291, 2023.

[24] Bo Qiao, Liqun Li, Xu Zhang, Shilin He, Yu Kang, Chaoyun Zhang, Fangkai Yang, Hang Dong, Jue Zhang, Lu Wang, et al. Taskweaver: A code-first agent framework. arXiv preprint arXiv:2311.17541, 2023.

[25] Zihan Liang, Yufei Ma, Ben Chen, Zhipeng Qian, Huangyu Dai, Lingtao Mao, Xuxin Zhang, Chenyi Lei, and Wenwu Ou. Ig-search: Step-level information gain rewards for search-augmented reasoning. arXiv preprint arXiv:2604.15148, 2026.

[26] LangChain. The anatomy of an agent harness. Engineering blog, 2026. URL `https://blog.langchain.com/the-anatomy-of-an-agent-harness/`. Published: 2026-03-10. Accessed: 2026-03-12.

[27] Wenhao Wang, Peizhi Niu, Zhao Xu, Zhaoyu Chen, Jian Du, Yaxin Du, Xianghe Pang, Keduan Huang, Yanfeng Wang, Qiang Yan, et al. Mcp-flow: Facilitating llm agents to master real-world, diverse and scaling mcp tools. arXiv preprint arXiv:2510.24284, 2025.

[28] Zhuoxuan Jiang, Tianyang Zhang, Haotian Zhang, Yinong Xun, Yang Liu, Dehua Feng, Wen Si, and Shaohua Zhang. Llma4itops: A lightweight llm-based multi-agent framework for it operations and maintenance. In CCF International Conference on Natural Language Processing and Chinese Computing, pages 471–482. Springer, 2025.

[29] Pei Yang, Wanyi Chen, Yuxi Zheng, Xueqian Li, Xiang Li, Haoqin Tu, Jie Xiao, Yifan Pang, Bill Shi, Lynn Ai, et al. Aoi: Turning failed trajectories into training signals for autonomous cloud diagnosis. arXiv preprint arXiv:2603.03378, 2026.

[30] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems, 33:9459–9474, 2020.

[31] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In The Twelfth International Conference on Learning Representations, 2023.

[32] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 7036–7050, 2024.

[33] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In Proceedings of the AAAI conference on artificial intelligence, volume 38, pages 19724–19731, 2024.

[34] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. Advances in neural information processing systems, 36:8634–8652, 2023.

[35] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In Proceedings of the 36th annual acm symposium on user interface software and technology, pages 1–22, 2023.

[36] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. arXiv preprint arXiv:2502.12110, 2025.

[37] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. Advances in neural information processing systems, 36:46534–46594, 2023.

[38] Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models. arXiv preprint arXiv:2404.14387, 2024.

[39] Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. arXiv preprint arXiv:2602.02474, 2026.

[40] Yaolun Zhang, Yiran Wu, Yijiong Yu, Qingyun Wu, and Huazheng Wang. Live-evo: Online evolution of agentic memory from continuous feedback. arXiv preprint arXiv:2602.02369, 2026.

[41] Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. arXiv preprint arXiv:2511.20857, 2025.

A LLM Backbone Comparison

Original 36-event evaluation. To evaluate sensitivity to the underlying LLM, we compare seven models (of various versions and scalings) on the alert root-cause analysis task with a 36-event subset described in §3.1. Before each evaluation run, the Skill directory is reset to the clean seed state to eliminate cross-run contamination. All models use temperature T = 0.3. We find that, frontier models cluster tightly at the top and clearly separate from smaller ones: DeepSeek-V3.2, GLM5, and Qwen3.5-35B achieve pass@1 within a 5.6-point band (77.8%, 77.8%, and 72.2% respectively); DeepSeek-V3.2 is the only model reaching 100% at pass@5. While dropping from 35B to 9B costs 25% of pass@1 (72.2% $\rightarrow$ 47.2%), and further reductions to 4B and 0.8B cause catastrophic degradation (36.1% and 13.9%). Practical recommendation is to use a backbone of around 35B parameters or larger for online deployment of BIAN QUE; smaller models exhibit substantially less robust behavior on operational reasoning tasks.

Table 8: Original backbone comparison on 36 events. pass@k is reported as correct cases (percent).

Model	pass@1	pass@2	pass@3	pass@4	pass@5
GLM5	28 (77.8%)	30 (83.3%)	32 (88.9%)	33 (91.7%)	35 (97.2%)
DeepSeek-V3.2	28 (77.8%)	31 (86.1%)	33 (91.7%)	35 (97.2%)	36 (100.0%)
Qwen3.5-35B-FP8	26 (72.2%)	29 (80.6%)	32 (88.9%)	34 (94.4%)	34 (94.4%)
Qwen3.5-27B-FP8	23 (63.9%)	25 (69.4%)	29 (80.6%)	30 (83.3%)	32 (88.9%)
Qwen3.5-9B	17 (47.2%)	22 (61.1%)	26 (72.2%)	28 (77.8%)	29 (80.6%)
Qwen3.5-4B	13 (36.1%)	15 (41.7%)	18 (50.0%)	20 (55.6%)	21 (58.3%)
Qwen3.5-0.8B	5 (13.9%)	6 (16.7%)	8 (22.2%)	8 (22.2%)	9 (25.0%)

Expanded 60-event evaluation. Every backbone was evaluated on the same 60 alert-RCA events in three independent runs, with identical snapshots, Skill and knowledge resources, temperature, budget, and evaluator. The Skill directory was reset before each run.

Table 9: Backbone comparison (percent; mean $\pm$ standard deviation).

Backbone	pass@1	pass@2	pass@3	pass@4	pass@5
DeepSeek-V4-Pro-preview	86.7 ± 3.3	90.6 ± 2.5	94.4 ± 1.0	97.2 ± 1.0	98.3 ± 1.7
GLM-5.2	80.0 ± 3.3	85.0 ± 1.7	90.0 ± 1.7	93.9 ± 1.0	96.7 ± 1.7
Qwen3.5-35B-FP8	71.7 ± 4.4	79.4 ± 3.5	85.6 ± 2.5	90.0 ± 1.7	93.3 ± 1.7
Qwen3.5-27B-FP8	63.9 ± 3.5	70.6 ± 2.5	80.0 ± 3.3	84.4 ± 2.5	89.4 ± 2.5
Qwen3.5-9B	47.2 ± 3.5	60.6 ± 2.5	70.6 ± 2.5	77.2 ± 2.5	81.1 ± 2.5
Qwen3.5-4B	37.2 ± 3.5	42.8 ± 2.5	50.0 ± 1.7	55.0 ± 1.7	58.3 ± 1.7
Qwen3.5-0.8B	13.3 ± 1.7	16.7 ± 1.7	20.6 ± 1.0	22.2 ± 1.0	25.0 ± 1.7

The expanded study updates the two frontier entries to DeepSeek-V4-Pro-preview and GLM-5.2 while retaining the Qwen family. It is a separate evaluation with more cases and repeated runs, not a numerical correction to the original 36-event table.

B Limitations and Future Work

Several directions remain open. First, the current framework covers the reasoning and diagnosis stages of operational workflows but does not automate the subsequent execution of remediation actions (e.g., performing rollbacks or scaling operations); extending the framework to closed-loop autonomous remediation is a natural next step. Second, while the Agent Matrix currently uses deterministic metadata-key lookup, incomplete metadata and unseen event types motivate more flexible future routing mechanisms. Third, the coordination across multiple agent executions within a single incident remains largely manual; developing orchestration protocols for multi-agent collaboration in complex, cascading failure scenarios presents an interesting research direction. Fourth, directly quantifying the compounding effect between the knowledge pathway and the Skill pathway requires a longer

evaluation horizon than our current six-month deployment permits (§2.4.3) and is left to future work. Finally, the residual RCA errors concentrate on newly deployed services for which the knowledge base has not yet accumulated sufficient experience; we expect the self-feedback mechanism to narrow this gap over time, but validating that hypothesis will require a multi-year evaluation.

The main deployment is observational, and the 13-day study is a single live trajectory with potential overlap between the evaluator and feedback provider. The six-case intervention does not evaluate transfer to future incidents. Private infrastructure and event records limit exact external reproduction. The newer cross-business deployments provide preliminary evidence rather than mature long-term validation.

C Repeated Evaluation and External Baselines

The Skill directory was reset to the same initial state before each run. Table 10 reports mean, standard deviation, and a run-level 95% confidence interval calculated across three runs using a t distribution. With only three runs, these intervals quantify run-to-run uncertainty but should not be interpreted as precise population intervals.

Table 10: Repeated 104-event evaluation.

Metric	Mean $\pm$ standard deviation	Run-level 95% CI
pass@1	$77.9 \pm 2.5\%$	$[71.7, 84.1]\%$
pass@2	$84.0 \pm 1.1\%$	$[81.3, 86.7]\%$
pass@3	$89.7 \pm 2.0\%$	$[84.7, 94.7]\%$
pass@4	$92.9 \pm 1.5\%$	$[89.2, 96.6]\%$
pass@5	$94.6 \pm 0.6\%$	$[93.1, 96.1]\%$

We compare BIAN QUE with: STATIC, manually authored Skills frozen throughout evaluation; Naive RAG, semantic top-k retrieval of source descriptions and knowledge; and a Voyager-style O&M adaptation that retrieves similar historical Skills, uses self-verification, and appends new Skills without revising existing ones. Knowledge-enabled methods share the same corpus and all methods share the LLM and budget.

Table 11: Overall results on 104 alert/inspection events (percent; mean $\pm$ standard deviation across three runs).

Method	pass@1	pass@2	pass@3	pass@4	pass@5
BIAN QUE	77.9 ± 2.5	84.0 ± 1.1	89.7 ± 2.0	92.9 ± 1.5	94.6 ± 0.6
STATIC	64.7 ± 1.1	71.2 ± 1.0	75.0 ± 1.7	77.6 ± 4.3	78.8 ± 4.8
Voyager-style + knowledge	55.6 ± 1.5	68.5 ± 1.8	75.5 ± 1.9	80.1 ± 2.0	83.3 ± 2.1
Voyager-style, no knowledge	48.2 ± 1.6	57.9 ± 2.0	63.1 ± 2.3	66.4 ± 2.5	68.6 ± 2.8
Naive RAG	41.3 ± 2.5	43.9 ± 1.5	47.1 ± 1.0	50.3 ± 1.1	53.2 ± 1.1

Table 11 reports overall results; Appendix C.1 gives the scenario breakdown. On alert events, BIAN QUE and the knowledge-enabled Voyager-style adaptation are close at pass@1 (68.2% versus 67.5%). The inspection difference is larger (85.0% versus 47.4%). This suggests that the task context affects the relative performance of the procedures. Because these methods differ in several aspects of retrieval and Skill maintenance, the comparison does not isolate deterministic routing or in-place refinement individually.

C.1 Scenario-Level Results

Tables 12 and 13 give the alert and inspection results for the 104-event evaluation. Entries are percentages (mean $\pm$ standard deviation across three runs). The two scenarios differ in their triggering information: an alert already specifies a symptom, whereas a scheduled inspection starts from a module context.

Table 12: Alert-event results ($N = 44$).

Method	pass@1	pass@2	pass@3	pass@4	pass@5
BIAN QUE	68.2 ± 3.9	79.5 ± 2.3	89.4 ± 2.6	93.9 ± 2.6	95.5 ± 0.0
STATIC	67.4 ± 7.3	70.5 ± 2.3	75.0 ± 2.3	76.5 ± 3.5	78.0 ± 4.7
Voyager-style + knowledge	67.5 ± 2.3	76.8 ± 2.4	81.3 ± 2.6	84.2 ± 2.7	86.0 ± 2.9
Voyager-style, no knowledge	65.9 ± 2.3	74.0 ± 2.6	77.6 ± 2.8	79.7 ± 3.1	81.0 ± 3.4
Naive RAG	45.5 ± 3.9	49.2 ± 1.3	53.0 ± 2.6	56.1 ± 3.5	59.1 ± 2.3

Table 13: Proactive-inspection results ($N = 60$).

Method	pass@1	pass@2	pass@3	pass@4	pass@5
BIAN QUE	85.0 ± 1.7	87.2 ± 1.0	90.0 ± 1.7	92.2 ± 1.0	93.9 ± 1.0
STATIC	62.8 ± 6.7	71.7 ± 2.9	75.0 ± 1.7	78.3 ± 6.0	79.4 ± 5.1
Voyager-style + knowledge	47.4 ± 2.0	62.8 ± 2.5	71.6 ± 2.6	77.3 ± 2.7	81.5 ± 2.8
Voyager-style, no knowledge	35.9 ± 1.9	46.8 ± 2.7	53.1 ± 3.1	57.2 ± 3.6	60.1 ± 4.0
Naive RAG	38.3 ± 1.7	40.0 ± 1.7	42.8 ± 1.0	46.1 ± 1.0	48.9 ± 1.0

D Release-Interception Evaluation

We constructed 57 real release-interception cases from the search deployment, each verified by on-call SREs. Every configuration used the same event snapshots, LLM, sampling settings, and evaluator. MAIN FLOW is full BIAN QUE; STATIC freezes the initial Skill pool; NOKNOW disables short- and long-term knowledge retrieval.

Table 14: Release-interception results by run.

Configuration	Run	pass@1	pass@2	pass@3	pass@4	pass@5
MAIN FLOW	1	45/57 (78.9%)	45/57 (78.9%)	46/57 (80.7%)	47/57 (82.5%)	47/57 (82.5%)
	2	47/57 (82.5%)	50/57 (87.7%)	51/57 (89.5%)	53/57 (93.0%)	53/57 (93.0%)
	3	48/57 (84.2%)	49/57 (85.9%)	49/57 (85.9%)	50/57 (87.7%)	51/57 (89.5%)
STATIC	1	42/57 (73.7%)	42/57 (73.7%)	42/57 (73.7%)	42/57 (73.7%)	43/57 (75.4%)
	2	45/57 (78.9%)	48/57 (84.2%)	49/57 (86.0%)	49/57 (86.0%)	50/57 (87.7%)
	3	43/57 (75.4%)	43/57 (75.4%)	44/57 (77.2%)	44/57 (77.2%)	44/57 (77.2%)
NOKNOW	1	42/57 (73.7%)	45/57 (78.9%)	46/57 (80.7%)	46/57 (80.7%)	46/57 (80.7%)
	2	42/57 (73.7%)	44/57 (77.2%)	45/57 (78.9%)	46/57 (80.7%)	47/57 (82.5%)
	3	40/57 (70.2%)	42/57 (73.7%)	43/57 (75.4%)	43/57 (75.4%)	44/57 (77.2%)

Table 15: Release-interception summary: percent, mean $\pm$ standard deviation across three runs.

Configuration	pass@1	pass@2	pass@3	pass@4	pass@5
MAIN FLOW	81.9 ± 2.7	84.2 ± 4.6	85.4 ± 4.4	87.7 ± 5.3	88.3 ± 5.4
STATIC	76.0 ± 2.7	77.8 ± 5.6	78.9 ± 6.3	78.9 ± 6.3	80.1 ± 6.6
NOKNOW	72.5 ± 2.0	76.6 ± 2.7	78.4 ± 2.7	78.9 ± 3.0	80.1 ± 2.7

E Feedback-Pathway Case Analysis

The intervention uses the six residual failures from the originally reported 104-event run. Each condition applies one pathway-specific update before sampling. The index j below identifies the first successful sample in its sequence; it is not a correction-round count or an estimated pass@j rate. The joint condition records success within five samples without an exact first-success index.

In the observed samples for Case F, only the joint condition succeeded. The Skill update added the missing `ksn_base_info` source for CPU and memory measurements. The knowledge update added an operational rule that missing observations must not be interpreted as evidence that an alert

Table 16: Case-level dual-pathway ablation.

Case	Knowledge only (Skill frozen)	Skill only (knowledge frozen)	Joint update
A	failure	failure	failure
B	$j = 1$	$j = 4$	success
C	$j = 1$	$j = 3$	success
D	$j = 4$	failure	success
E	$j = 2$	$j = 1$	success
F	failure	failure	success
Overall at pass@5	4/6 (66.7%)	3/6 (50.0%)	5/6 (83.3%)

is non-actionable. Neither single-path condition resolved the event; their combination did. This is targeted evidence on selected residual cases. There is no matched no-update resampling condition, and the experiment does not evaluate later unseen cases.

F Additional Production Evidence

F.1 Paired Production-Log Replay

A paired replay uses 42 recent real search alerts (12 actionable, 30 non-actionable). Each configuration sees the same snapshot in three runs. Full BIAN QUE reaches $85.7\pm6.3\%$ pass@1 and $96.8\pm1.4\%$ pass@5; removing Skill-mediated data retrieval yields $37.3\pm1.4\%$ and $42.9\pm2.4\%$, while removing knowledge yields $55.6\pm6.0\%$ and $86.5\pm3.6\%$. The matched snapshots test the effect of removing each context component in this replay. They do not isolate Skill routing from the benefit of access to additional data, nor replicate the live temporal study.

Table 17: Paired 42-alert production-log replay (percent; mean $\pm$ standard deviation, three runs).

Setting	pass@1	pass@2	pass@3	pass@4	pass@5
Full	85.7 ± 6.3	93.7 ± 1.4	96.0 ± 1.4	96.8 ± 1.4	96.8 ± 1.4
No Skill retrieval	37.3 ± 1.4	38.9 ± 1.4	39.7 ± 1.4	42.1 ± 3.6	42.9 ± 2.4
No knowledge	55.6 ± 6.0	71.4 ± 4.1	82.5 ± 6.9	84.9 ± 5.0	86.5 ± 3.6

Knowledge transition. Knowledge retrieval was disabled during June 15–21 and enabled during June 22–28 while the remaining system configuration was unchanged. The disabled week had 203 total cases and 90 ratings (44.3% coverage): 83 positive and 7 negative. The enabled week had 321 cases and 118 ratings (36.8% coverage): 116 positive and 2 negative. The rated-subset negative rate fell from 7.8% to 1.7% (two-sided Fisher exact $p = 0.042$). Because the weeks were not randomized and rating coverage differed, this is supporting in-situ evidence rather than a causal estimate over all events.

Live feedback study and annotation. During the 13-day study, adaptive and frozen arms processed every event from the same incoming stream and began with identical Skill and knowledge snapshots. Five on-call SREs rotated across days. Each case was normally labeled once by the SRE on duty using subsequent operational outcomes; that SRE could also provide corrective feedback. Conventional inter-annotator agreement is therefore unavailable. Senior SREs performed weekly spot-checks of handled cases and corrections. This differs from the offline benchmark, where three SREs independently agreed unanimously on 100/104 cases and jointly adjudicated four.

Safety policy. Only the human-facing notification is suppressed; the source alert remains active. Persistent events are re-evaluated, unhandled alerts are forwarded, and protected-metric incidents are escalated. P0 alerts, under 5% of volume, are never suppressed. No high-severity incident was recorded as missed or delayed during the six-month observation. Weekly audits found fewer than 1% of sampled agent-handled cases requiring reliability follow-up.

G Early Cross-Business Evaluation

Early deployments without architectural changes provide cross-business evidence: advertising proactive inspection solved 26/30 cases (86.7%); recommendation alert diagnosis solved 25/30 (83.3%); and recommendation release interception solved 26/34 (76.5%). These deployments ran for only about one month, so they support transfer feasibility rather than mature generalization.

Table 18: Early cross-business evidence after approximately one month.

Business line	Pattern	Correct/total	Accuracy
Advertising	Proactive inspection	26/30	86.7%
Recommendation	Alert diagnosis	25/30	83.3%
Recommendation	Release interception	26/34	76.5%

The core framework is reused across business lines. Platform connectors must be adapted to the target infrastructure, while business-specific Skills and knowledge are generated and updated after deployment. Counts from these studies should not be summed to infer unique incidents without checking overlap.

Ethics and Reproducibility Statements

Ethics. BIAN QUE is deployed as an assistive O&M system with human oversight. Only human-facing notifications are suppressed; source alerts remain active, persistent and unhandled issues are escalated, and P0 alerts always reach the on-call SRE. The evaluated system does not autonomously execute remediation. Production records are proprietary and are reported in aggregate. Broader impacts are discussed after the experiments.

Code and reproducibility. The public implementation is available at `https://github.com/benchen4395/BianQue_Assistant`. The repository provides a Skill-execution workflow, dependency and setup instructions, example configurations, and a mock-data demo; production data-source and knowledge-store integrations must be supplied by users. It is not a complete release of the internal production system or an exact reproduction package for the reported experiments. Section 4 describes the implementation and evaluation protocol; Appendices A–G report extended model comparisons, repeated runs, baseline definitions, release-interception results, pathway interventions, and production procedures. Proprietary event records and private infrastructure limit exact external reproduction.

AI use. Generative AI was used to assist with manuscript preparation. All AI-assisted content is subject to human review and verification.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction describe the operational paradigm, Flexible Skill Arrangement, and feedback mechanism, with evidence in Sections 2 and 3. Autonomous pass@k is distinguished from SRE-assisted cumulative accuracy, and observational production results are not presented as randomized causal estimates.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix B discusses remediation, routing, coordination, evaluation-horizon, and reproduction limitations. Appendices C, F, and G further qualify uncertainty, evaluator/feedback overlap, and preliminary cross-business evidence.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper presents an applied operational framework rather than theoretical results. Its equations describe architecture and composition, not theorems requiring proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 and Appendices A–G describe the backbone, sampling settings, event composition, reset protocol, metrics, baselines, and evaluation procedures; the public workflow implementation is linked in the Ethics and Reproducibility Statements. These support reimplementation of the framework, while exact numerical reproduction remains limited by proprietary production records and private infrastructure.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code, setup instructions, example configurations, and a mock-data demo are public at `https://github.com/benchen4395/BianQue_Assistant`. Proprietary production datasets and internal integrations are not released, and the repository does not provide a complete package for faithfully reproducing all reported experimental results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 specifies inference settings, dataset composition, metrics, ablations, and reset procedures; Appendices A, C, D, and F supply additional model, baseline, repeated-run, and production details. No model training or fine-tuning is performed.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix C reports means, sample standard deviations, and run-level 95% t confidence intervals over three independent runs, with explicit small-sample limitations. Appendices D and F report additional repeated-run variation; the latter reports a two-sided Fisher exact test for the rated-subset comparison, while the original single-run and observational deployment summaries remain point estimates.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Section 4 identifies a single NVIDIA Tesla L20 GPU for the default Qwen3.5-35B-FP8 backbone and describes the serving stack. Per-experiment wall-clock time, memory use, total compute, and resource details for every compared backbone are not fully reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: As stated in the original submission, the research conforms to the NeurIPS Code of Ethics: it uses operational telemetry rather than end-user personal data, and feedback is provided by internal SREs in their regular duties. The Ethics and Reproducibility Statements and Appendix F describe human oversight and deployment safeguards.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The Broader Impacts discussion covers reduced operational toil and improved reliability alongside delayed-response risks, over-reliance, erosion of expertise, and confidentiality concerns. It describes the evaluated safeguards and their limits, with additional deployment details in Appendix F.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release newly trained model weights or proprietary production datasets. The public asset is workflow code with example configurations and mock data; operational deployment safeguards are described in Appendix F.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: Third-party models and prior work are cited in the paper, and no model weights are redistributed. However, the manuscript does not comprehensively list the exact licenses and terms for all model versions, libraries, and serving components used, so a complete affirmative licensing claim is not made.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The public repository documents the released workflow code, dependencies, setup, integration interfaces, and mock-data demo: `https://github.com/benchen4395/BianQue_Assistant`. The Ethics and Reproducibility Statements distinguish this release from the internal production system and disclose the unavailability of proprietary datasets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: As in the original submission, the study does not involve crowdsourcing or research with human subjects; annotations and feedback are supplied by internal professional SREs as part of routine operational duties. Evaluation and annotation procedures are described in Section 4 and Appendix F.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The original submission reports that this work does not involve human-subjects research or collection of personal or sensitive participant data. Engineer feedback is part of routine operational work; no new IRB approval or exemption is asserted here.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are central to reasoning, Skill generation and update, and knowledge distillation in Section 2. Section 4 specifies the default backbone and inference settings, and Appendix A compares alternative backbones.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.