ActionNex: A Virtual Outage Manager for Cloud Computing

Zhenfeng Lin[*]
Ming Hao[†]
Ryan Zhang[*]
Ze Li[‡]
Chetan Bansal[§]
Murali Chintalapati[‡]
Salman Zafar[*]
Haoji Hu[‡]
Xuchao Zhang[§]
Junhao Li [*]
Oleg Kulygin [*]
Hatay Tuna [†]
Sheila Jiang [*]
Angie Anderson [*]

Abstract

Outage management in large-scale cloud operations remains heavily manual, requiring rapid triage, cross-team coordination, and experience-driven decisions under partial observability. We present **ActionNex**, a production-grade agentic system that supports end-to-end outage assistance, including real-time updates, knowledge distillation, and role- and stage-conditioned next-best action recommendations. ActionNex ingests multimodal operational signals (e.g., outage content, telemetry, and human communications) and compresses them into critical events that represent meaningful state transitions. It couples this perception layer with a hierarchical memory subsystem: long-term Key–Condition–Action (KCA) knowledge distilled from playbooks and historical executions, episodic memory of prior outages, and working memory of the live context. A reasoning agent aligns current critical events to preconditions, retrieves relevant memories, and generates actionable recommendations; executed human actions serve as an implicit feedback signal to enable continual self-evolution in a human–agent hybrid system. We evaluate ActionNex on eight real Azure outages (8M tokens, ~4,000 critical events) using two complementary ground-truth action sets, achieving 71.4% precision and 52.8–54.8% recall. The system has been piloted in production and has received positive early feedback.

Keywords

aiops, llm, agentic, hierarchical memory, long-term memory, action recommendation, multimodal, continual learning, self-evolving

[*]Microsoft PRIMO, Redmond, USA.
[†]Microsoft Azure CTO Office, Redmond, USA.
[‡]Microsoft Azure Core, Redmond, USA.
[§]Microsoft Research, Redmond, USA.

1 Introduction

Cloud computing has became dominant computing paradigm and continues to expand in both scale and complexity. Outage Management is a core function in large-scale cloud operations: it coordinates detection, assessment, investigation, mitigation, and resolution to minimize service disruptions and uphold strict service-level objectives. Yet, in sharp contrast to the growing automation of software creation, outage-management operations remain heavily manual—relying on human-driven triage, cross-team coordination, and experience-based decision making under time pressure.

In the recent years, the rapid gains in LLM reasoning have enabled agentic systems to plan over long horizons, use external tools, operate under partial observability, and improve through feedback [14], [19], [10], [6], [18]. As a result, automating outage-management tasks is becoming increasingly reachable. Multiple systems have been proposed in this domain [17], [4], [8], [5], [11], [7], [12]. Most of them, however, focus on one or a few aspects of outage-management activities—such as troubleshooting execution, triage, or causal reasoning.

In practice, outage management spans a much wider and more variable operational space. During large-scale outages, hundreds of responders across distinct roles may participate; the response can last days; and dozens of work-streams often run in parallel while exchanging information intensively and continuously. Modern outage-management environments therefore involve a broad range of cognitive and coordination-heavy tasks, including custom-impact analysis, monitoring and diagnosis, triage, internal and external communications, cross-service recovery orchestration, resolution validation, and so on. As a result, specialized systems typically provide only partial assistance in the real-world operations.

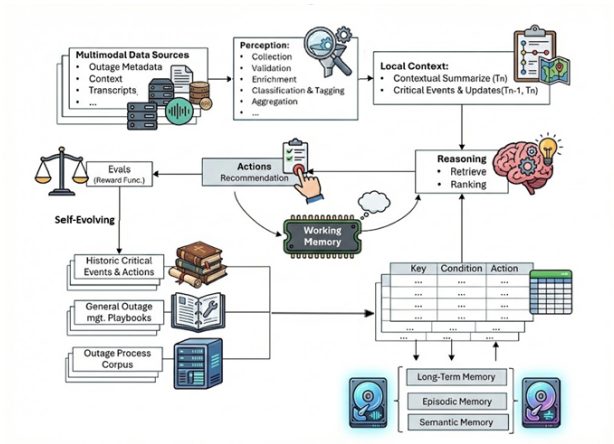

Figure 1: ActionNex Framework

To design a general system that can tackle these tasks end-to-end, we turn to reinforcement learning paradigm and present **ActionNex**, a continual, self-evolving outage management solution which uses action as general abstraction for knowledge, decision-making and tasks.

The core idea is to learn what actions to take and when to take the action from human experiences while operating in a human-agent hybrid system so that it can recommend actions in a timely fashion (and execute them in the future).

In real time, ActionNex autonomously ingests multi-source signals—including structured data from outage systems, bridge meeting transcripts, groups chats, and other operational artifacts—and distills into actionable knowledge through hierarchical abstraction in its memory. Conditioned on outage context, stage, and role, it proposes next-best actions. Meanwhile, the actions ultimately taken by humans act as an implicit reward signal, enabling to continuously learn, adapt, and improve over time.

The architecture of ActionNex is shown at Fig. 1. It consists of following key components:

- `Mutlimodal Data Perception Layer`: Ingests multimodal signals from diverse sources and encodes them into critical events that compactly represent and summarize outage symptoms and state transitions.
- `Knowledge Memory Layer`: Stores the extracted knowledge as *(key, condition, action)* entries derived from historical human operations and playbooks. It continuously accumulates new knowledge extracted from outage management processes and self-evolves.
- `Reasoning Agent Layer`: Applies policy and/or learned reasoning to align current critical events with the preconditions and adaptively generate most likely next-best actions recommendation.

We validate ActionNex with real outage data from Azure, demonstrating the effectiveness of our continual self-evolution design. Specifically, we extract ground-truth actions from raw operational traces in offline batches using latest LLM, and manually label for quality assurance. We then replay traces to trigger action recommendations, which are evaluated against the ground truth over subsequent time windows. The system has been deployed in production piloting and has received positive early feedbacks.

In summary, our contributions include:

(1) We design and deploy a general production-grade agentic outage management system that supports Q&A, real-time updates, knowledge distillation, and Key-Condition-Action (**KCA**) based real-time recommendations.
(2) We propose a novel self-evolving hierarchical memory subsystem.
(3) We evaluate the system with real Azure production outage data and demonstrate its effectiveness.

2 Mutlimodal Data Perception Layer

The Multimodal Data Perception Layer constitutes the front-end of the ActionNex. It transforms diverse, high-volume signals into a human-readable compact representation – critical events – for downstream reasoning and action recommendation instead of exposing noisy raw input directly to the reasoning agent. These events summarize the system state and the outage investigation evolution. This transformation not only improves the detection of operationally salient state changes, but also allow the whole process to be interpretable and easy to maintain.

2.1 Input Modalities

The system ingests multiple complementary data modalities that jointly characterize the evolving state of outage:

- Outage-management operational metadata (e.g., affected services/regions, timestamps, severity indicators, structured status updates, communications).
- Human communication artifacts (bridge meeting transcripts, chats, operator notes revealing hypotheses, decisions, ownership, and implicit intent).
- Telemetry-derived contextual signals (alerts, anomaly annotations, partner events reflecting system changes and mitigation effects).

Each modality is a partial and noisy on its own, so effective action recommendation requires joint interpretation across modalities rather than isolated signal processing.

2.2 Normalization and Perception

Incoming signals are first processed through a perception pipeline that emphasizes semantic alignment over raw fidelity. Its output is then used to generate critical events.

First, inputs are normalized into a common representation, validated for temporal consistency and source authenticity, and deduplicated to remove redundant observations. Next, the system enriches signals by extracting entities (e.g., services, regions, components, teams), resolving aliases, and augmenting observations with dependency and topology context. Signals are tagged and classified by outage phase, event type, severity, and relevance to support focused downstream reasoning. Finally, cross-modalities signals are aggregated into coherent contextual states representing the system's current understanding of the outage.

This pipeline deliberately avoids early decision-making. Its reduces heterogeneity while preserving the semantic content needed to reason about operational impact.

2.3 Critical Event Abstraction

Reasoning directly over raw multimodal outage data is neither scalable nor aligned with operational practice. Signals are high-volume, noisy, and often redundant, and decisions are rarely triggered by individual alerts; they respond to meaningful changes in system state. We therefore use critical events as principled abstraction of state of change. By collapsing observations into explicit state transitions, the perception layer reduces noise, captures temporal structure, and aligns perception with human outage-manager mental models.

This layer outputs a sequence of critical events—observations or transitions that materially alter outage understanding or control (e.g., scope expansion, mitigation start/complete, dependency recovery, probable root cause identification). The system computes deltas between consecutive time windows and promotes significant deltas to critical events, storing them in short-term memory as the interface to downstream reasoning.

Although critical events are a lossy abstraction, they are sufficient for action recommendation because they retain decision-critical state changes (e.g., mitigation progress, dependency recovery, blast-radius expansion), are derived from aggregated cross-modal enriched context (topology, life-cycle phase, corroboration), and provide the predicates required by ActionNex's KCA reasoning without exposing raw noise.

3 Knowledge Memory Layer

The goal of memory design is to enable effectively information learning, indexing, and retrieval for relevant action recommendations. There are a large body of research demonstrating the essential role memory plays for self-evolving [3], [2], [13], [9], [1], [15]. Our knowledge memory layer is a hieratically structure consisting of multiple memory blocks shown in the table 1 which capture complementary forms of context (e.g., recent outage state, historical trajectories, and reusable operational knowledge).

These components together accumulate knowledge and guide recommendations. Due to space constraints, we will give a brief introduction on working and episodic memory first and focus on our core component—long-term memory afterwards. Working memory tracks the live outage state (recent actions, results, stage) to prevent redundant or conflicting steps. Episodic memory stores past outage cases (critical events, decisions, outcomes) to reuse proven strategies and avoid failures. Over time, both support continuous improvement.

The main functionality of long-term memory is to connect perceived outage dynamics to operational actions by organizing knowledge into structured key, condition, action (KCA) units for retrieval, condition evaluation, and policy grounding. Because KCA elements are stored as text, they are interpretable and easy for domain experts to audit and correct; domain experts can also directly author KCA entries, enabling learning from both real-world outage data and human knowledge. This design lets the reasoning agent match the current outage state against accumulated experience and recommend actions aligned with operational practice. The KCA definitions are shown below:

Condition: Summarizes generalized symptoms distilled from critical events (e.g., root cause confirmed, mitigation in progress, dependency recovered, or escalation triggered). Each condition acts as a stable semantic identifier that consolidates multiple observations of the same operational phenomenon.

Key: Provides the contextual scopes for interpreting a condition, typically tied to life-cycle stage, service domain, or responsibility boundary (e.g., investigate/mitigate/recover). Keys and conditions are embedded and indexed; either may surface KCA candidates first, with the other used for filtering or ranking, enabling robust retrieval under partial or uncertain context.

Action: Specifies the operational response once a KCA unit is retrieved and its context is deemed relevant. Actions are not primary retrieval targets during recommendation reasoning progress; they are normalized from historical executions and established procedures, represented as parameterized semantic templates that adapt to specific services/regions/roles while preserving real operator structure for precision, auditability, and trust.

In operation, each KCA unit functions as a contextual implication: under a given key, if its condition is supported

Memory	Source & Content	Role in Reasoning
Long-Term Memory (Knowledge Base)	Aggregated reference knowledge, including outage management manuals, SOPs, service **playbooks**, guidelines, and domain-specific expertise, indexed as *Key–Condition–Action* (KCA) triples.	Provides **canonical solutions** and best-practice actions for known scenarios when matching conditions are satisfied.
Episodic Memory (Case Library)	Logs of **historical outages**, including critical events, decisions taken, and outcomes (success or failure). Each case records the outage context (e.g., service, severity), the identified precondition, and the actions used for resolution.	Provides **analogous examples** from prior outages. Successful actions suggest viable strategies, while failed actions indicate approaches to avoid.
Working Memory (Current Context)	**Live Outage state**, including recently attempted actions, intermediate results, conversation history, and the current outage stage or status. This memory is continuously updated as the outage evolves.	Provides **situational awareness**, ensuring recommendations account for what has already occurred, thereby preventing redundant or contradictory actions and maintaining a coherent action sequence.

Table 1: Memory types used by the agent and their roles in outage reasoning and action recommendation.

by the evolving critical events, the linked action becomes a candidate recommendation. The reasoning agent embeds the current outage state and retrieves relevant units via similarity of keys or conditions, then incorporates the remaining context to refine relevance, rank alternatives, and suppress unsafe or mistimed responses. Because selection is retrieval-driven rather than dependent on exact rule triggering, the framework naturally accommodates partial evidence, ambiguity, and incremental updates as new information arrives. Executed outcomes are continuously written back into memory, enabling the knowledge base to evolve while remaining anchored to canonical and auditable forms.

Overall, KCA provides a principled bridge between perception and decision-making, balancing abstraction with flexibility while preserving alignment with how human outage managers interpret situations and authorize actions.

All memory components support continual updates implementing implicit self-evolution. In online fashion, the system selectively promotes working memory traces into episodic memory, and periodically summarizes new knowledge and merge into long-term memory. In offline fashion, domain experts can optionally review memory records and provide update recommendations, which are incorporated to evolve the whole memory system.

4 Reasoning Agent Layer

The Reasoning Agent Layer is the decision-making core of ActionNex. It bridges perceived critical events and the memory layer (Section 3) to produce action recommendations. This layer interprets incoming critical events to infer the current outage situation as a precondition. It then combines this inferred context with retrieved long-term, episodic, and working memories to recommend the next action under its policy.

Example

04:19Z: *BridgeVoice: "thermal protection… triggered shutdowns" of "storage and direct drive," while "cooling… restored" and "temperatures… stabilizing" under "SEV-1." The system matched the Resolve precondition for "Drive Core Platform Recovery Sequence."*
04:22Z: *recommended "controlled power-up and recovery sequence" for the "shutdown direct drive cluster."*
04:37Z: *humans executed "Controlled, staged recovery initiated" and "completed… recovery sequencing plans."*

Step 1 Align Events to Preconditions. The agent converts raw outage events at time t into a structured precondition describing the current situation, along with meta-context, such as outage stage and affected service.

Step 2 Retrieve Relevant Memories. The agent retrieves the top-k entries from long-term memory and the top-m similar cases from episodic memory, incorporating working-memory context when available.

Step 3 Adaptive Action Recommendation with Feedback. Given the precondition and retrieved memories, the agent formulates a set of candidate actions and uses its policy—implemented via an LLM prompt—to recommend the next best action. If the LLM determines that the available context is insufficient (e.g., missing key signals or conflicting evidence), the agent autonomously re-enters Step 2, refining its memory queries or expanding the retrieval scope to gather additional relevant information. This refine process is similar to the refine process in React [16] to fine-tune the prompt and regenerate the recommendation results. Prompt 1 illustrates the action recommendation step.

The next section will evaluate how this reasoning approach performs in practice and how it interacts with the

System. You are a reasoning agent that recommends outage mitigation actions based on structured context and operational knowledge.

Situation (Precondition). Power and cooling have been restored after automatic thermal-protection shutdowns impacting storage and direct-drive components. All dependencies are verified and mitigation is complete (*Stage*: Mitigate; *Outage*: "High Thermal Triggered Shutdowns").

Recent Observations (Working Memory). At 04:19Z, bridge updates reported thermal-triggered shutdowns. Subsequent communications confirmed cooling restoration and temperature stabilization. Outage severity is SEV−1.

Relevant Knowledge (Long-Term).

- *Runbook*: After thermal shutdowns, verify environmental stability before initiating the **Drive Core Platform Recovery Sequence**.
- *KCA*: Ensure mitigation and dependency health prior to controlled service restoration.

Similar Past Outages (Episodic Memory).

- *Outage #312* (Aug 2024): Thermal event caused storage shutdowns. **Action**: Cooling restored, followed by staged platform recovery.
- *Outage #198* (Jan 2023): Protective shutdowns due to high temperature. **Action**: Environmental stabilization before recovery.

Task. Given the current precondition, observations, and knowledge, **recommend the next mitigation or recovery action**.

Prompt 1: Simplified prompt for one action recommendation attempt

rest of the system (including any learning or feedback mechanisms for refining the policy).

5 Evaluation Results

We evaluate recommendation quality, system efficiency, and contribution of key components. Our evaluation is designed to reflect both the noise of real outage response and alignment with high-confidence operational guidance.

Dataset Overview: We curated a dataset consisting of eight real-world outages, comprising approximate 8M tokens of contextual data, including outage, operational logs, meeting transcripts, and group chats. From raw contexts, we extracted 4,374 critical events, serving as the intermediate state representation for downstream reasoning and led to 361 recommended actions. In parallel, 447 actions were independently identified and confirmed by human investigators and treated as ground-truth labels, without reference to the model predictions. Five outages were used for memory construction and development, while the remaining three were held out exclusively for evaluation.

LLM Models usage: Throughout this work, we employ **GPT-5.2** for information extraction, memory construction, and multi-step reasoning. We use **text-embedding-3-large** for embedding generation and memory indexing. For final prediction—ground-truth evaluation, we rely on **GPT-5.2 Thinking**, which provides higher-fidelity judgment for outcome assessment.

5.1 Ground Truth Construction

To enable reliable evaluation, we curate two complementary ground truth sets. We further invited domain experts to manually evaluate them and general feedback is that they cover almost all the common actions for outage management activities ignoring outage specific details.

G1 (In-the-wild actions): G1 consists of actions extracted directly from outage context. This set is closest to real deployment conditions, but noisy and occasionally ambiguous, as actions may be discussed without execution or logged without explicit attribution.

G2 (Traceable guidance actions): G2 is derived by filtering G1 to retain only actions explicitly supported by authoritative playbooks or handbooks. Relative to G1, G2 is more concise and easier to verify, focusing on higher-confidence, repeatable operational patterns.

Evaluating against both G1 and G2 allows us to disentangle two questions: (I) Robustness under realistic operational noise (G1), and (II) Correctness with respect to canonical, high-confidence guidance (G2).

5.2 Data Quality and Semantic Coverage

To validate the semantic quality of both ground-truth sets and ensure meaningful metrics, we perform a coverage analysis using embedding-space visualization. For each outage, we embed predicted and ground truth actions with outage life-cycle stage tags, and visualize them using t-SNE projections. Figure 2 shows predicted actions (square), G1 actions (circle) and playbook-extracted long-term memory actions (triangle).

Both sets, predicted actions broadly and evenly cover the same semantic regions occupied by other actions. This indicates that (I) the curated ground truth sets occupy coherent semantic regions rather than isolated artifacts, and (II) ActionNex generates a diverse yet relevant set of actions aligned with human operational actions. This supports that the reported precision/recall reflects real semantic alignment rather than extraction bias.

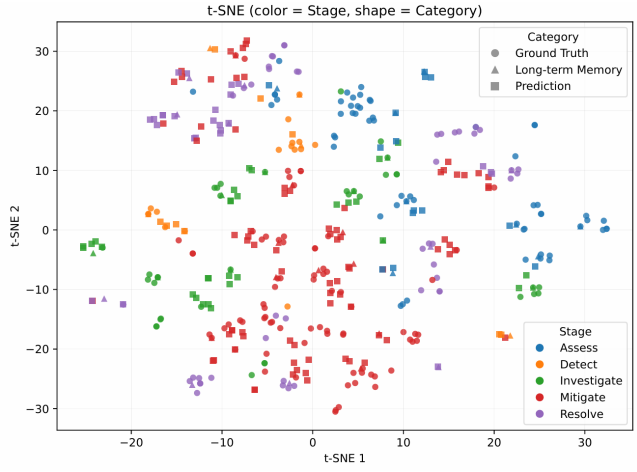

Figure 2: t-SNE view of actions

Ground Truth	Precision (%)	Recall (%)
G1	**71.4**	**52.8**
G2	71.4	54.8

Table 2: Accuracy for G1 and G2.

5.3 Performance

Table 2 summarizes precision and recall evaluated against two ground-truth sets, **G1** and **G2**. Overall, results suggest ActionNex can learn operational actions from human experiences and playbooks and surface them at appropriate time given evolving outage context. Precision happen to be identical for both cases because matching ground truth are preserved when generating **G2** from **G1**. While this precision might appear modest, manual inspection reveals that majority of false positives actually are operationally reasonable recommendations that were simply not captured or promoted into ground truth. Relatively low recall is mainly because limited early stage outage context delays confident triggering, or related conditions are not identified by the system yet. ActionNex's ability to incorporate execution outcomes and feedback over time provides a natural path to improving both coverage and timing as additional outage are observed.

To understand recommendation quality evolves over time, we analyze performance across five outage stages. Figure 3 shows *Stage-wise precision and recall*. Recall increases while precision decreases as outage progresses. Early stages contain spares and uncertain signals, so recommendations are constrained by incomplete context and hypotheses. As additional critical events are confirmed and outcomes become more observable, preconditions can be inferred more reliably, expanding coverage and improving recall. From product perspective, this pattern propose a stage-adaptive strategy—for

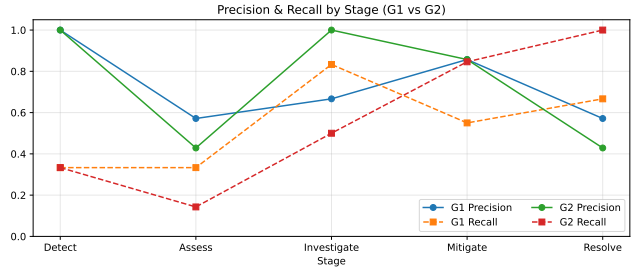

Figure 3: Accuracy over Stages for G1 and G2

instance, *emphasizing episodic-memory analogies in early stages to improve recall, and shifting toward stricter condition gating later to preserve precision.*

6 Conclusions

We presented a production-grade agentic system for outage management that combines multimodal context, critical-events, and action recommendations with a hierarchical memory layer centered on long-term Key–Condition–Action and continual learning from human actions. On real production outages, ActionNex demonstrates strong recommendation quality and practical utility in a human-agent workflow. Future work will improve early-stage coverage, strengthen safety for online learning, and measure impact on operational outcomes such as time-to-mitigation at scale.

References

[1] Anonymous. 2026. REMem: Reasoning with Episodic Memory in Language Agents. In *ICLR*.

[2] Abrar et al. Anwar. 2024. ReMEmbR: Long-Horizon Spatio-Temporal Memory for Robot Navigation. *arXiv:2409.13682* (2024).

[3] Zouying et al. Cao. 2025. Remember Me, Refine Me: Dynamic Procedural Memory for Agent Evolution. *arXiv:2512.10696* (2025).

[4] Ruowei et al. Fu. 2025. OncallX: LLM-Powered Multi-Agent Collaboration for On-Call Automation. In *ASE*.

[5] Pouya et al. Hamadanian. 2023. A Holistic View of AI-Driven Network Incident Management. In *HotNets*.

[6] Bowen et al. Jin. 2025. Search-R1: Training LLMs to Reason with Search via Reinforcement Learning. *arXiv:2503.09516* (2025).

[7] Jian-Guang et al. Lou. 2013. Software Analytics for Incident Management of Online Services. In *ASE*.

[8] Jiacheng et al. Mao. 2025. Agentic Troubleshooting Guide Automation for Incident Management. *arXiv:2510.10074* (2025).

[9] Siru et al. Ouyang. 2025. ReasoningBank: Scaling Agent Self-Evolution with Reasoning Memory. *arXiv:2509.25140* (2025).

[10] Timo et al. Schick. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv:2302.04761* (2023).

[11] Chenxu et al. Wang. 2025. Towards LLM-Based Failure Localization in Production Networks. In *SIGCOMM*.

[12] Zefan et al. Wang. 2024. RCAgent: Cloud Root Cause Analysis with Autonomous Agents. In *CIKM*.

[13] Tianxin et al. Wei. 2025. Evo-Memory: Benchmarking Test-Time Learning with Self-Evolving Memory. *arXiv:2511.20857* (2025).

[14] Tianxin et al. Wei. 2026. Agentic Reasoning for Large Language Models. *arXiv:2601.12538* (2026).

[15] Menglin et al. Xia. 2026. Memora: A Harmonic Memory Representation Balancing Abstraction and Specificity. *arXiv:2602.03315* (2026).

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

[17] Zhaoyang et al. Yu. 2025. Triangle: Empowering Incident Triage with Multi-Agent Systems. In *ASE*.

[18] Aohan et al. Zeng. 2023. AgentTuning: Enabling Generalized Agent Abilities for LLMs. *arXiv:2310.12823* (2023).

[19] Wentao et al. Zhang. 2025. AgentOrchestra: Orchestrating Multi-Agent Intelligence. *arXiv:2506.12508* (2025).