MTTR-A: Measuring Cognitive Recovery Latency in Multi-Agent Systems

Barak Or
Masiv AI
barak@masivai.com
barakorr@gmail.com

Abstract

Ensuring cognitive stability in autonomous multi-agent systems (MAS) is a central challenge for large-scale, distributed AI. While existing observability tools monitor system outputs, they cannot quantify how rapidly agentic workflows recover once reasoning coherence has been lost. We adapt classical reliability metrics-*Mean Time-to-Recovery (MTTR)*, *Mean Time Between Failures (MTBF)*, and related ratios-into the cognitive domain, defining **MTTR-A (Mean Time-to-Recovery for Agentic Systems)** as a runtime measure of cognitive recovery latency. MTTR-A quantifies the time required for a MAS to detect reasoning drift and restore consistent operation, capturing the *recovery of reasoning coherence* rather than infrastructural repair.

A benchmark simulation using the AG News corpus and the LangGraph orchestration framework was conducted, modeling recovery latencies across multiple reflex modes. Automated reflexes restored stability within approximately 6 s on average, while human-approval interventions required about 12 s. Across 200 runs, the median simulated MTTR-A was 6.21 ± 2.14 s, MTBF $\approx 6.73 \pm 2.14$ s, and NRR ≈ 0.08, demonstrating measurable runtime resilience across reflex strategies.

By formalizing recovery latency as a quantifiable property of distributed reasoning-and deriving reliability bounds linking recovery time and cognitive uptime-this work establishes a foundation for **runtime dependability in agentic cognition**, transforming cognitive recovery from an ad-hoc process into a standardized, interpretable performance metric.[1]

Keywords: multi-agent systems, cognitive reliability, agentic orchestration, LLM agents, runtime stability, fault tolerance, AI governance, reflex control, LangGraph

1. Introduction

Classical system engineering defines reliability through metrics such as availability, mean time between failures (MTBF), and mean time to recovery (MTTR) [1]. These indicators were developed for deterministic infrastructures-servers, networks, or control loops-where recovery denotes restoring a failed component to service. In contrast, failures in modern *agentic* AI systems are often cognitive: agents may continue running while reasoning drifts, memories desynchronize, or coordination deteriorates.

As multi-agent systems (MAS) built on large language models (LLMs) become increasingly autonomous and interconnected, such cognitive faults cannot be captured by conventional observability or uptime metrics [2, 3]. Recent surveys highlight similar reliability and coordination challenges in LLM-based MAS, underscoring the need for runtime governance and recovery mechanisms [4, 5].

[1]Code, data, and experimental configurations are available at https://github.com/BarakOr1/MTTR-A.

A system may remain functionally "up" while its reasoning has diverged or its safety policy degraded. This exposes a fundamental gap: how can we measure *recovery* not of infrastructure, but of cognition itself?

To our knowledge, no prior orchestration framework defines a quantitative runtime metric for cognitive recovery; existing approaches focus on observability or schema enforcement, not on recovery latency itself [6, 7]. Industry reports concentrate on model capability and alignment rather than orchestration-level recovery, highlighting the absence of runtime reliability metrics in production AI systems [8–10].

Motivation for System-Level Metricization: In realistic deployments, cognitive or coordination failures are inevitable. Rather than analyzing each agent in isolation, reliability must be evaluated at the level of the entire orchestration -the MAS as an interacting intelligence network. We therefore adapt the classical MTTR metric into an orchestration-level cognitive context, defining **MTTR-A (Mean Time-to-Recovery for Agentic Systems)** as a measure of how long a distributed reasoning system takes to detect drift and restore coherence. This perspective treats the multi-agent architecture as a single cognitive organism whose communication and recovery pathways determine runtime resilience [11]. The practical objective, as in classical dependability engineering, is to *minimize* MTTR-A through improved synchronization, reflex coordination, and governance policies.

Central Claim

Cognitive faults in multi-agent systems are inevitable; what defines reliability is not their absence but how quickly the system detects, isolates, and recovers from them.

Real-world MAS increasingly operate in environments where recovery latency determines safety, performance, and stability. In autonomous vehicle fleets, the agents are self-driving vehicles that communicate through vehicle-to-vehicle and infrastructure networks; even a two-second delay in recovering from a coordination or perception fault can trigger cascading collisions or traffic paralysis. In aerial drone swarms, the agents are autonomous UAVs performing collaborative missions such as mapping or rescue; prolonged recovery from signal loss or reasoning drift can fragment formation and terminate the mission. In industrial robotic cells, the agents are networked manipulators and sensors executing synchronized assembly tasks; delayed resynchronization after a fault can halt production or cause mechanical conflict. In distributed financial trading systems, the agents are autonomous trading algorithms interacting through shared market data; slow recovery from an erroneous strategy or drifted policy can amplify instability and cause systemic loss. In cybersecurity defense networks, the agents are detection and classification models monitoring distributed endpoints; long recovery from false positives or misclassifications can block legitimate activity and reduce coverage. Finally, in modern LLM-based orchestration frameworks the agents are reasoning modules coordinating tool use, memory, and planning; high cognitive recovery time leads to inconsistent outputs and degraded trust. Across these domains, the reliability of the MAS depends on preventing every fault and on how rapidly the system detects, isolates, and restores coherent operation once a fault occurs.

In practical enterprise deployments, such orchestration-level dependability requires not only measurement but also structured runtime control. When agent workflows drift, loop, or violate safety rules, the system must detect, interrupt, and safely recover-leaving an auditable trace of reasoning and actions. This motivates the emergence of what can be described as a *Cognitive Reliability* layer for distributed AI systems. Such layer may provide: Open Cognitive Telemetry (OCT), Causal Drift Graph, Trust, Drift, and Recovery Metrics, Policy Hooks and Enforcement, and Audit & Compliance Interface.

- **Open Cognitive Telemetry (OCT):** standardized tracing of reasoning steps across agents.
- **Causal Drift Graph:** identification of coordination drift or role creep.
- **Trust, Drift, and Recovery Metrics:** continuous estimation of runtime cognitive stability.
- **Policy Hooks and Enforcement:** pause mechanisms, quarantine, or rollback rogue agents.
- **Audit and Compliance Interface:** signed traces, SLO reporting, and post-incident replay.

Figure 1: Key operational modules of the Cognitive Reliability Layer, providing observability, drift control, and runtime governance for multi-agent orchestration.

These operational elements illustrate the broader system context in which MTTR-A becomes essential, as it provides a measure for cognitive recovery and runtime reliability in large-scale MAS. This is a measurable basis for runtime resilience in distributed reasoning systems. Importantly, MTTR-A evaluates recovery behavior at the orchestration layer without requiring visibility into the internal reasoning of each agent or the semantics of their communications. It assumes that cognitive faults are inevitable in open, distributed systems, and focuses instead on how quickly the orchestration loop detects, isolates, and restores coherence-treating recovery latency itself as the measurable indicator of cognitive reliability.

This work situates reliability engineering within the domain of agentic cognition, linking classical fault-tolerance principles with reflexive control architectures for LLM-based MAS. Combined with a taxonomy of recovery reflexes and an empirical LangGraph benchmark, establishes a foundation for measuring and improving cognitive reliability in MAS. The empirical benchmark employs a real-text retrieval environment based on the AG News dataset to ensure that reasoning-drift and recovery cycles are observed under natural language conditions.

1.1. Our Approach: A Reliability Metricization Framework

This paper positions MTTR-A as a reliability metric for MAS. Rather than evaluating semantic accuracy or reasoning quality, we focus on *runtime dependability* -how quickly and predictably a distributed agent workflow detects, isolates, and recovers from cognitive faults. It unifies a reflex taxonomy, a latency-based metric family (MTTR-A, MTBF, NRR), and a LangGraph-based telemetry pipeline into a reproducible methodology for benchmarking agentic reliability.

System Engineering Perspective

Each agent acts as a repairable subsystem, and the orchestration layer composes these into a hierarchical reliability model where reflex actions represent components, agents represent subsystems, and the entire multi-agent workflow represents the system.

As illustrated in Figure 2, this hierarchical reliability structure treats each agent as a subsystem composed of reflexive recovery components (rollback, replan, retry, approve), while the orchestration layer governs overall system-level dependability.

1.2. Contributions

This work makes four main contributions:

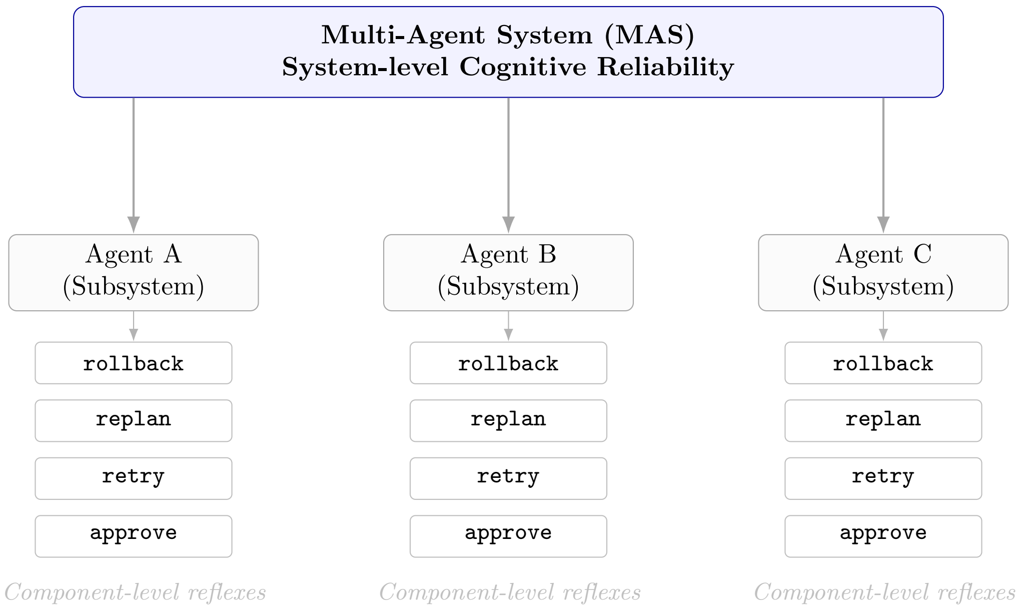

Figure 2: Hierarchical reliability model (illustrative). Each agent acts as a subsystem composed of reflexive recovery components (rollback, replan, retry, approve). For simplicity, three representative agents (A-C) are shown; the approach generalizes to any number of agents.

1. **Adaptation of classical reliability metrics to cognitive orchestration:** We extend established dependability measures-MTTR, MTBF, and related ratios into the cognitive domain, defining MTTR-A as a runtime metric for reasoning recovery in multi-agent systems.
2. **A taxonomy of runtime recovery reflexes:** We systematize reflexive control actions into five categories:-recovery, human-in-the-loop, governance, coordination, and safety-providing a standardized vocabulary for cognitive fault management and orchestration behavior.
3. **An empirical LangGraph simulation for cognitive reliability:** We implement and release a LangGraph-based experiment (200 runs) that evaluates reasoning drift, recovery latency, and normalized reliability (NRR) across reflex modes.
4. **Formal reliability theorems:** We derive two theoretical results linking runtime metrics to long-run cognitive uptime. First, under an alternating-renewal model [12], we prove that NRR_{sys} provides a conservative lower bound on the steady-state cognitive uptime (Theorem 1). Second, we extend this relation to include recovery-time variance, establishing a confidence-aware lower bound that incorporates uncertainty in cognitive recovery (Theorem 2).

2. Related Work

LLM Agents and Deliberation: Chain-of-Thought [13], ReAct [14], Toolformer [15], and Tree-of-Thoughts [16] provide prompting, acting, and tool-use patterns for single agents. Beyond single-agent prompting, multi-agent lines-Generative Agents [17], CAMEL [18], Reflexion [19], and Voyager [20]-together with recent surveys [7, 21] map collaboration benefits and failure modes (e.g., coordination errors, unsafe tool calls, memory divergence). These works primarily improve *ex-ante* deliberation and coordination; they do not quantify *ex-post* recovery latency or runtime dependability once reasoning has drifted.

Recent studies have expanded MAS collaboration paradigms through reflection and debate mechanisms -such as Reflective Multi-Agent Collaboration [22], Chain-of-Agents [23], MAGIS [24], Multi-LLM Debate [25], and DigiRL for device-control agents [26] - as well as empirical analyses on data-scale effects for UI-control agents [27]. While these frameworks enhance coordination quality and communication efficiency, they remain performance-oriented and do not formalize runtime dependability or recovery-latency metrics as addressed in the present work.

Resilient and Secure MAS: Decades of MAS work study consensus, fault tolerance, and security: resilient consensus under DoS and Byzantine settings [28, 29], fault-tolerant control [30–33], adaptive/secure consensus [28], and broader resilient modelling [11]. These works provide metrics, models, and control laws but do not standardize *runtime recovery primitives* for LLM-centric agent orchestration. Comprehensive reviews such as [4, 5] have catalogued the rapid development of LLM-based multi-agent frameworks and identified persistent challenges in coordination, memory consistency, and agent orchestration. These analyses reinforce the motivation for a quantitative dependability metric - directly measures recovery dynamics rather than task performance or interaction quality.

Gap in Existing Frameworks

Observability explains *what happened*; resilience metrics quantify *how severe it was*; but orchestration reliability determines *what to do next*-how the system should recover, replan, seek human approval, record an audit snapshot, or execute within a controlled sandbox.

Existing orchestration frameworks such as Guardrails and AutoGen partially address reliability but operate at different layers. Guardrails enforces schema and safety validation *after* text generation, while AutoGen structures multi-agent conversations and role interactions. Neither defines runtime recovery reflexes once coordination faults or reasoning drift occur. We introduce a control vocabulary that acts *within the execution loop*-linking telemetry, policy, and reflex actions such as rollback or sandbox-execute.

Recent orchestration frameworks such as AutoGen [2], OpenAgents [6], and emerging control-plane studies including ALAS [34] and Advancing Agentic Systems [35] extend LLM coordination to multi-agent workflows, but they primarily address communication and execution graphs rather than quantitative recovery metrics such as MTTR-A.

Complementary benchmarking efforts such as Spider2-V [36] evaluate multimodal professional-workflow agents, yet omit explicit reliability or recovery-time indicators, underscoring the absence of quantitative runtime dependability benchmarks in current agent evaluations.

3. Method

This section formalizes the orchestration control method underlying the MTTR-A framework. We propose a reflex-oriented architecture for runtime recovery in MAS, where each cognitive fault triggers an automatic or human-mediated reflex action. The architecture defines five families of recovery reflexes and organizes them within a closed feedback loop linking telemetry, meta-monitoring, and execution (Figure 3, Table 1). Each recovery cycle produces measurable latency components from which MTTR-A and related dependability metrics are computed (see Section 4).

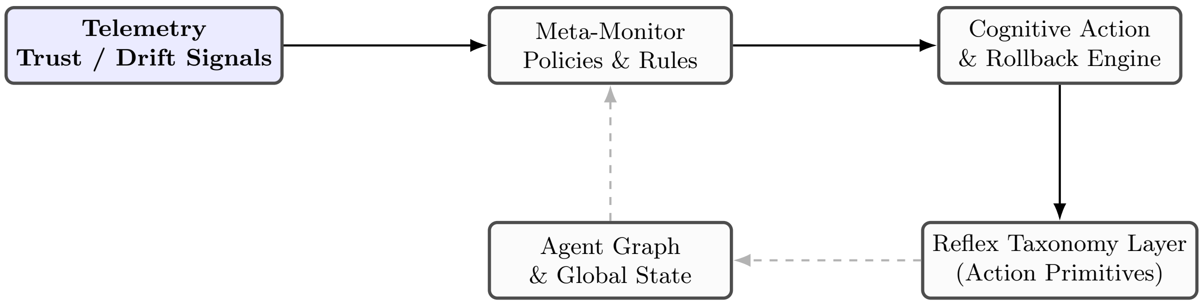

Figure 3: Closed-loop control architecture linking telemetry, policy monitoring, action execution, and global state feedback. Arrows represent the data flow between perception (telemetry), decision (meta-monitor), and action (execution engine), forming a reflex-based feedback system for the Multi-Agent System (MAS).

3.1. Reflex Taxonomy and Control Loop

Each reflex family is defined by a *trigger*, a *policy*, and an *effect on system state*. Telemetry streams supply signals such as trust degradation, drift detection, or compliance violations, which activate the appropriate reflex within the control loop. The overall pipeline forms a reflex-based feedback system that continuously monitors cognition, detects deviations, and enforces corrective actions.

Table 1: Reflex taxonomy and corresponding recovery actions. Each category defines runtime mechanisms for detection, intervention, and safe continuation within the orchestration control loop of the MAS.

Category	**Action**	**Purpose**	**Typical Trigger**
Recovery	auto-replan, rollback, tool-retry, fallback-policy, safe-mode	Regenerate plans, restore checkpoints, or retry failed tools using simplified policies and minimal verified behaviors.	Planning failure, drift event, tool timeout, repeated failure, or safety constraint.
Human-in-the-Loop	human-approve, human-override, human-review, escalate-to-expert	Introduce manual oversight, approval, or escalation to experts for compliance-critical or ambiguous cases.	Compliance or high-impact step, anomaly detected, or specialized task requiring human judgment.
Governance	auto-diagnose, self-heal, confidence-gate, vote-or-consensus, sandbox-execute, drift-rollback	Perform self-diagnosis, pause and validate, isolate risky actions, or revert models when drift or uncertainty is detected.	Anomaly pattern, recoverable environment fault, low confidence, divergent outcomes, or drift threshold exceeded.
Coordination	broadcast-update, negotiate-task, sync-state, lock/release-resource	Maintain distributed coherence across the MAS through shared context propagation, task negotiation, and resource synchronization.	Global context change, workload imbalance, context divergence, or resource contention.
Safety	graceful-abort, force-terminate, audit-snapshot	Ensure controlled or emergency shutdown with full trace capture for audit and recovery verification.	Controlled shutdown condition, unrecoverable error, or compliance requirement.

Behavioral Scope: The reflex families in Table 1 define complementary layers of recovery control, from automatic replanning to human-in-the-loop oversight and safety shutdown. Collectively, they ensure that cognitive drift, tool faults, and coordination errors are corrected without systemic collapse.

3.2. Runtime Policy and Coordination Dynamics

Within a MAS, recovery latency is shaped by the type of reflex invoked and also by the efficiency of policy selection and inter-agent coordination. The meta-monitor chooses tselects the fastest valid recovery action based on telemetry triggers, such as tool errors, confidence loss, or detected drift, while policy compliance. Agents coordinate through a shared global context, synchronizing memory and actions during recovery to prevent cascading inconsistencies. These decision and communication latencies collectively define the measured MTTR-A: the time interval from fault detection to complete restoration of coherent operation across the MAS.

The decision process that governs reflex selection is depicted in Figure 4, which maps fault triggers-such as tool errors, low confidence, or reasoning drift-to their corresponding recovery actions within the control loop.

Reducing MTTR-A therefore depends on minimizing both the policy-selection delay and the coordination overhead that emerge during distributed recovery.

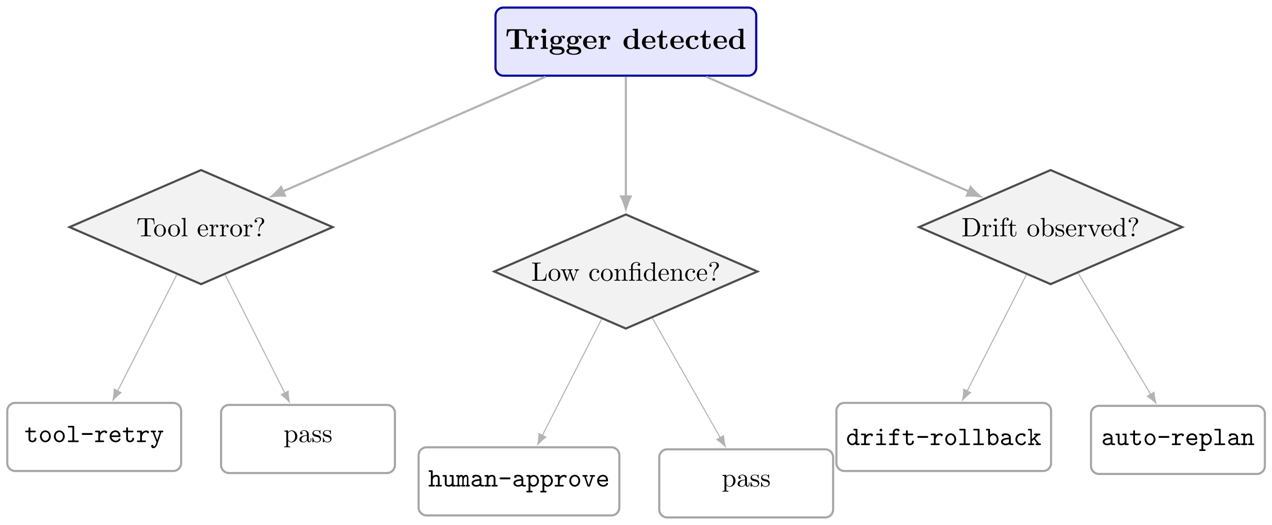

Figure 4: Policy decision tree used to select reflex actions. Branches correspond to fault types (tool error, low confidence, or drift) and map each to specific recovery primitives such as `tool-retry` or `drift-rollback`.

4. Experiment and Evaluation Metrics

This section presents both the theoretical framework and the experimental configuration used to evaluate the proposed runtime reliability metrics for MAS. First, we establish the conceptual motivation and formal definitions of MTTR-A and related measures that quantify cognitive recovery and dependability. Then, we describe the experimental setup and data pipeline used to empirically compute these metrics in a simulated MAS built with the LangGraph framework.

4.1. Evaluation Metrics and Cognitive Dependability Context

The proposed reliability metrics were evaluated using the LangGraph framework, which enables stateful, graph-based MAS workflows built on LLMs. The experimental setup simulated reasoning drift, reflex activation, and recovery cycles across multiple interacting agents, allowing quantitative assessment of runtime cognitive dependability.

We draw directly from classical dependability theory [1], which formalizes reliability using MTTR, MTBF, and availability ratios. In classical terms, MTTR measures the expected duration between system failure and full restoration of service under a binary up-down model. In contrast, cognitive systems such as MAS operate continuously, with faults that are semantic, distributed, and partially recoverable. We therefore reinterpret these dependability metrics at the level of cognitive orchestration, quantifying the efficiency with which a MAS detects, isolates, and restores reasoning coherence following a coordination or drift event.

4.2. Formal Reliability Metricization Framework

We adapt classical dependability measures to the MAS domain, defining *MTTR-A* (Mean Time-to-Recovery for Agentic Systems) as a runtime measure of cognitive recovery latency. While traditional MTTR quantifies infrastructural repair time, MTTR-A captures the temporal efficiency of *cognitive recovery*-the process by which a distributed reasoning network identifies drift and restores coherent operation.

Key Metric Definitions

The framework defines three core reliability metrics: *MTTR-A* measures how quickly reasoning recovers after drift, *MTBF* captures the average stable interval between faults, and *NRR* combines both into a single indicator of runtime reliability.

System-of-Agents Model: We model the orchestrated workflow as a **system of agents** $\mathcal{S} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_N\}$, where each agent $\mathcal{A}_i$ functions as an autonomous *subsystem* with its own local reasoning, detection, and recovery dynamics. Each subsystem executes internal reflex components c_{ik} (e.g., `tool-retry`, `auto-replan`, `rollback`, `human-approve`) that act as *components* of the overall recovery architecture. Thus, MAS reliability is described hierarchically:

- **Component level:** reflex-mode recovery latency ($MTTR\text{-}A_{\text{mode}}$)
- **Subsystem level:** per-agent metrics ($MTTR\text{-}A_i$, $MTBF_i$)
- **System level:** aggregate orchestration metrics ($MTTR\text{-}A_{sys}$, $MTBF_{sys}$, NRR_{sys})

This hierarchy mirrors systems engineering principles, where each subsystem contributes independently to overall reliability.

Formal Definition: Let $\{(t_{f,i}^{(j)}, t_{r,i}^{(j)})\}_{j=1}^{M_i}$ denote the set of M_i fault-recovery episodes for agent $\mathcal{A}_i$, where $t_{f,i}^{(j)}$ is the timestamp at which a cognitive fault is detected and $t_{r,i}^{(j)}$ is when reasoning coherence is re-established. The local recovery duration is:

$$\Delta T_i^{(j)} = t_{r,i}^{(j)} - t_{f,i}^{(j)}. \qquad (1)$$

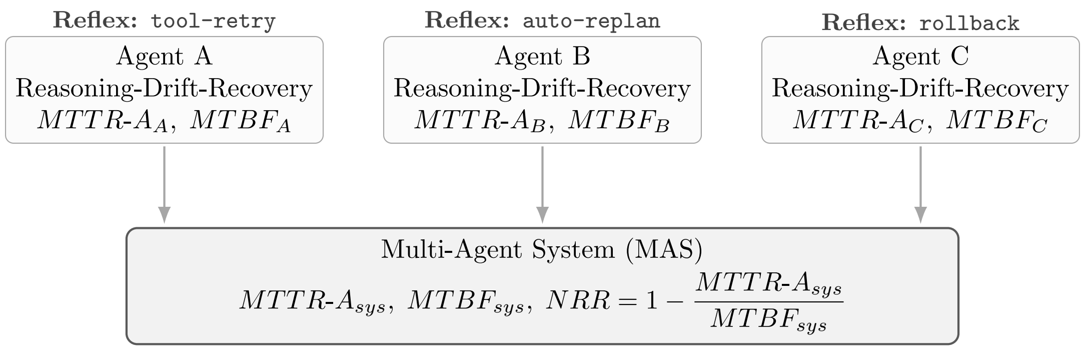

Figure 5: Hierarchical reliability model for a MAS. Agents A-C represent reasoning subsystems, each characterized by local recovery and stability metrics. System-level dependability emerges from the aggregation of agent-level performance.

The per-agent **Mean Time-To-Recovery** is then:

$$\text{MTTR-A}_i = \frac{1}{M_i} \sum_{j=1}^{M_i} \left(t_{r,i}^{(j)} - t_{f,i}^{(j)} \right) = \mathbb{E}[\Delta T_i], \tag{2}$$

and the system-level value, aggregating across N agents, is defined as:

$$\text{MTTR-A}_{sys} = \frac{1}{N} \sum_{i=1}^{N} \text{MTTR-A}_i. \tag{3}$$

Robust Estimator: As recovery-time distributions in MAS are typically right-skewed due to occasional long-latency interventions (e.g., human approvals), we also suggest using a robust estimator, the **Median Time-To-Recovery**:

$$\text{MedTTR-A}_{\text{sys}} = \text{median}_{1 \le j \le M} \left[\Delta T^{(j)} \right]. \tag{4}$$

This mitigates the influence of rare but extreme events and complements MTTR-A as a distribution-invariant measure.

Latency Decomposition: Each recovery episode, at either agent or system level, can be decomposed into additive latency components:

$$\Delta T^{(j)} = T_{\text{detect}}^{(j)} + T_{\text{decide}}^{(j)} + T_{\text{execute}}^{(j)}, \tag{5}$$

where T_{detect} is the drift detection latency, T_{decide} is the policy-selection delay, and T_{execute} is the reflex execution time.

Complementary Metrics: Two additional dependability metrics complete the MAS reliability framework:

1. **Mean Time Between Cognitive Faults (MTBF):** average stable duration between drift events:

$$\text{MTBF}_i = \frac{1}{M_i} \sum_{j=1}^{M_i} \left(t_{\text{fault},i}^{(j)} - t_{\text{recovered},i}^{(j-1)} \right), \tag{6}$$

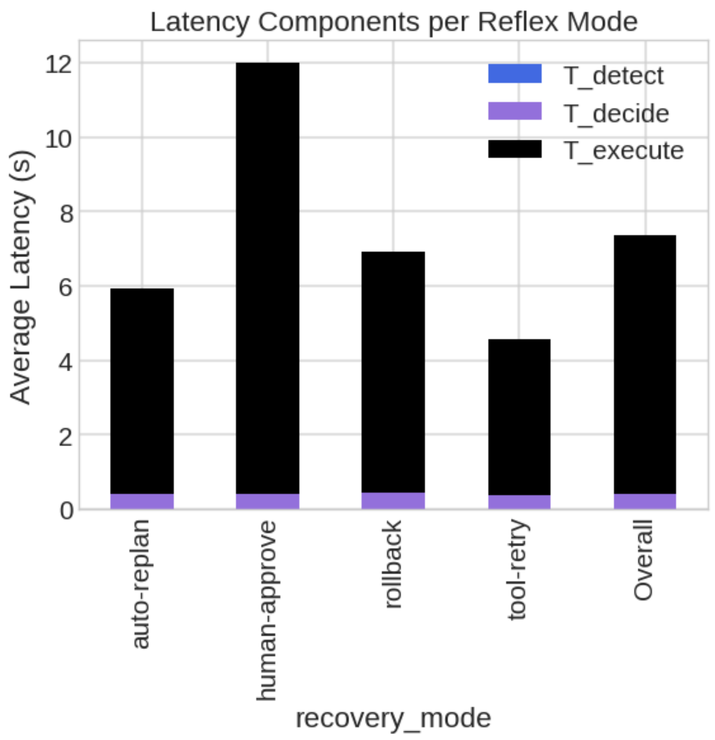

Figure 6: **Illustrative decomposition of MTTR-A.** Conceptual example showing the relative contributions of detection (T_{detect}), decision (T_{decide}), and execution (T_{execute}) latencies as defined in Equation (5). This figure illustrates the theoretical structure of recovery latency.

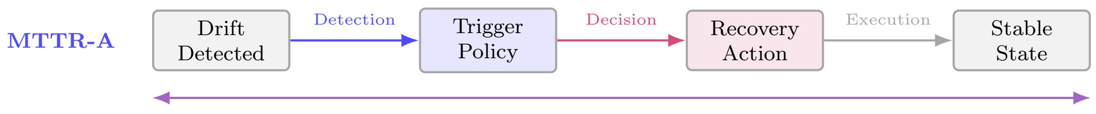

Figure 7: Compact MTTR-A timeline showing detection (blue), decision (purple), and execution (gray) phases leading to recovery. The horizontal arrow below represents the total MTTR-A duration.

and the system-level MTBF:

$$\text{MTBF}_{sys} = \frac{1}{N}\sum_{i=1}^{N} \text{MTBF}_i. \tag{7}$$

2. **NRR:** a dimensionless runtime reliability index:

$$\text{NRR}_{sys} = 1 - \frac{\text{MTTR-A}_{sys}}{\text{MTBF}_{sys}}. \tag{8}$$

   Values near 1 indicate MAS that recover rapidly relative to fault rate, while values approaching 0 or negative imply unstable recovery behavior.

We adapt the classical MTBF-MTTR relationship into a normalized, runtime reliability indicator for cognitive orchestration/ MAS, termed the *Normalized Recovery Ratio (NRR)*. While the underlying form derives from traditional availability approximations, its application here is newly adopted: the NRR is directly measurable from reasoning-drift and recovery events in MAS, providing a practical indicator of cognitive uptime during runtime operation.

We next formalize this relationship under the alternating-renewal model of dependability theory [1], showing that NRR_{sys} acts as a conservative lower bound on the system's long-run cognitive uptime.

Definition 1 (Steady-State Fraction of Cognitive Uptime). *Let the MAS operate as an alternating sequence of cognitively stable (*up*) and recovery (*down*) periods over time. Denote by $A_{\text{up}}(T)$ the total duration of coherent reasoning within the time interval $[0, T]$. The* steady-state fraction of cognitive uptime *is defined as the long-run ratio*

$$\pi_{\text{up},sys} := \lim_{T\to\infty} \frac{A_{\text{up}}(T)}{T}, \tag{9}$$

whenever the limit exists. It represents the asymptotic proportion of time during which the MAS maintains reasoning coherence, analogous to classical system availability.

Theorem 1 (NRR lower-bounds steady-state cognitive uptime). *Under the alternating-renewal model of dependability theory, the steady-state fraction of cognitive uptime in a MAS satisfies*

$$\pi_{\text{up},sys} = \frac{\text{MTBF}_{sys}}{\text{MTBF}_{sys} + \text{MTTR}_{sys}} = \frac{1}{1 + \lambda_{sys}\mu_{sys}} \geq 1 - \lambda_{sys}\mu_{sys} = \text{NRR}_{sys}, \tag{10}$$

where $\lambda_{sys} = 1/\text{MTBF}_{sys}$ and $\mu_{sys} = \text{MTTR}_{sys}$. Therefore, the normalized recovery ratio NRR_{sys} provides a conservative lower bound on the system's steady-state cognitive uptime. The inequality follows from the elementary bound $(1+x)^{-1} \geq 1-x$ for all $x \geq 0$. A proof is provided in Appendix A.

Definition 2 (Confidence-aware Normalized Recovery Ratio (NRR_α)). *Let R denote the random recovery duration of the MAS, with mean $\mu = \mathbb{E}[R] = \text{MTTR-A}_{sys}$ and standard deviation $\sigma = \sqrt{\text{Var}[R]}$. For any confidence level $\alpha \in (0, 1)$, define*

$$k_\alpha = \sqrt{\frac{1-\alpha}{\alpha}}, \qquad R_\alpha := \mu + k_\alpha \sigma.$$

Then, the confidence-aware normalized recovery ratio *is*

$$\text{NRR}_\alpha := 1 - \lambda_{sys} R_\alpha, \qquad \textit{where } \lambda_{sys} = 1/\text{MTBF}_{sys}. \tag{11}$$

This extends the classical NRR_{sys} by explicitly incorporating the variance of recovery times, providing a confidence-graded lower bound on runtime cognitive uptime.

Theorem 2 (Variance-aware lower bound on cognitive uptime). *Under the same alternating-renewal model of dependability theory, for any confidence level $\alpha \in (0, 1)$ the steady-state fraction of cognitive uptime satisfies*

$$\pi_{\text{up},sys} \geq 1 - \lambda_{sys}(\mu + k_\alpha \sigma) = \text{NRR}_\alpha, \tag{12}$$

with probability at least α. A complete proof is provided in Appendix A.

The adapted metrics MTTR-A_i, MedTTR-A_i, MTBF_i, and NRR_i (for each agent), and their system-level counterparts MTTR-A_{sys}, MTBF_{sys}, and NRR_{sys}, establish a unified quantitative foundation for evaluating *runtime cognitive dependability* in MAS.

4.3. Experimental Setup

The experimental evaluation employed LangGraph to simulate cognitive drift, reflex activation, and recovery across interacting agents. The following configuration was used.

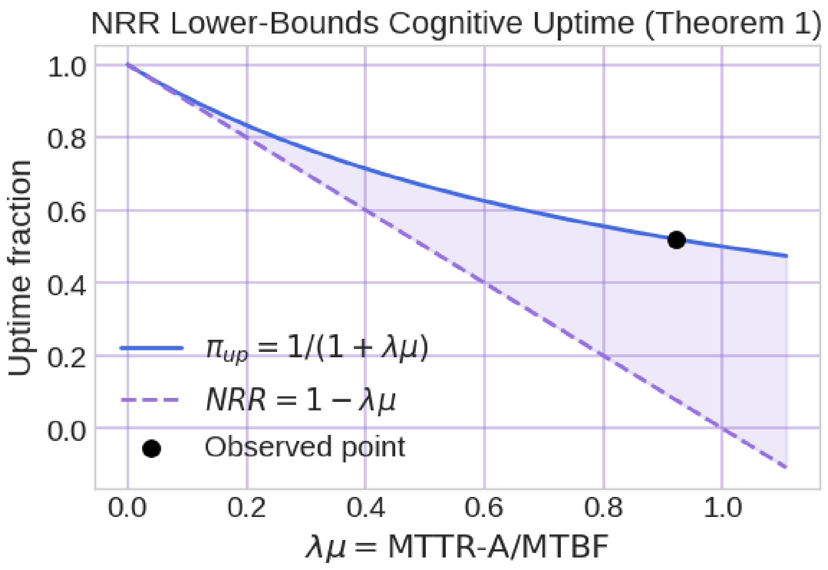

Figure 8: **NRR and variance-aware bounds on cognitive uptime.** The blue curve shows the analytical steady-state uptime $\pi_{\mathrm{up},sys} = 1/(1 + \lambda_{sys}\mu_{sys})$, the dashed purple line shows the first-order approximation $\mathrm{NRR}_{sys} = 1 - \lambda_{sys}\mu_{sys}$, and the gray shaded band represents the confidence-adjusted bounds NRR_α from Theorem 2.

4.4. Algorithmic Procedure for Measuring MTTR-A

Algorithm 1 defines a general procedure for measuring cognitive reliability metrics in multi-agent systems. It is agnostic to the underlying framework or dataset and can be applied to any orchestrated workflow that supports fault detection and recovery tracing. In our experiments, this procedure was instantiated using the LangGraph framework and the AG News corpus to simulate reasoning-drift-recovery cycles.

Algorithm 1 General Procedure for Computing Cognitive Reliability Metrics in Multi-Agent Systems

```
Require: Multi-agent system $\mathcal{S}$, fault detection policy $\pi_{\text{detect}}$, recovery reflex set $\mathcal{R}$, number
    of iterations $N$.
Ensure: System-level MTTR-A$_{sys}$, MTBF$_{sys}$, NRR$_{sys}$.
 1: for $i = 1$ to $N$ do
 2:     Observe system state; apply $\pi_{\text{detect}}$ to determine fault occurrence.
 3:     if fault detected then
 4:         Trigger appropriate recovery reflex $r \in \mathcal{R}$
 5:         Measure detection, decision, and execution latencies
 6:         Compute total recovery time $\Delta T_i = T_{\text{detect}} + T_{\text{decide}} + T_{\text{execute}}$
 7:     end if
 8: end for
 9: Estimate MTTR-A$_{sys}$ = median($\Delta T_i$)
10: Estimate MTBF$_{sys}$ from inter-fault intervals
11: Compute NRR$_{sys} = 1 - \frac{\text{MTTR-A}_{sys}}{\text{MTBF}_{sys}}$
12: return {MTTR-A$_{sys}$, MTBF$_{sys}$, NRR$_{sys}$}
```

In our implementation, Algorithm 1 was instantiated in the LangGraph orchestration framework as a three-node workflow representing the reasoning-drift-recovery cycle. The nodes operated sequentially as follows:

(1) `reasoning_node` retrieved documents from the AG News corpus and computed a retrieval confidence score,

$$c = \cos(\mathbf{q}, \mathbf{d}_{\text{top}}) = \frac{\mathbf{q} \cdot \mathbf{d}_{\text{top}}}{\|\mathbf{q}\| \|\mathbf{d}_{\text{top}}\|}, \tag{13}$$

where $\mathbf{q}$ is the query vector and $\mathbf{d}_{\text{top}}$ is the highest-scoring document.

(2) `check_drift_node` compared c to the drift threshold $\tau_{\text{drift}} = 0.6$ and flagged a cognitive fault whenever $c < \tau_{\text{drift}}$ (or via small stochastic perturbations).

(3) `recovery_node` executed one of four reflex modes- `auto-replan`, `rollback`, `tool-retry`, or `human-approve`-selected according to weighted policy sampling and simulated decision and execution latencies.

All state transitions were logged as structured telemetry (`.jsonl`) for offline computation of MTTR-A, MTBF, and NRR. Across 200 runs, timestamps for detection, decision, and execution were recorded, yielding per-mode recovery latencies and system-level dependability metrics.

4.5. Interpretation

This simulation reproduces distributed reasoning scenarios where autonomous agents collaborate under uncertainty. In such environments, $MTTR - A_{sys}$ captures the MAS's operational resilience, how rapidly reasoning coherence is restored after drift, providing a practical runtime indicator of cognitive reliability. Further implementation details, including computational environment, query pool, and configuration parameters, are provided in Appendix C.

5. Results and Discussion

The overall MedTTR-A across 200 LangGraph runs on the AG News dataset was 6.21 s ± 2.14 s (90th percentile: 10.73 s), indicating that typical cognitive recovery occurs within roughly six seconds after drift detection. The MTBF was 6.73 s ± 2.14 s, yielding an NRR of 0.077. These results confirm that the orchestration maintained stable recovery cycles with approximately 8% of runtime devoted to corrective action and no evidence of cascading instability.

These metrics indicate that the agentic workflow maintained stable recovery cycles with roughly 10% of runtime spent on corrective actions and no simulated evidence of cascading instability. Per-mode results are summarized in Table 2.

Table 2: Experimental results on the AG News dataset (N = 200 runs). Each value represents the median recovery time per reflex mode.

Recovery Mode	**MedTTR-A (s)**	**Std (s)**	**P90 (s)**	**Count**
auto-replan	5.94	0.70	6.81	93
tool-retry	4.46	0.61	5.40	42
rollback	6.99	0.43	7.45	44
human-approve	12.22	0.68	12.77	21

Distribution and Recovery Behavior

Figure 9 summarizes both the overall distribution and per-mode latency comparison. The results follow a right-skewed profile centered near 6-7 s, with most automated recoveries completing rapidly and a small tail corresponding to human oversight. This shape reflects high orchestration responsiveness and low variance across the experiment. Fully automated reflexes (`tool-retry`, `auto-replan`) restored stability within 5-7 s on average, while `rollback` operations showed slightly higher median times but reduced variability. The `human-approve` mode remained the slowest due to manual gating.

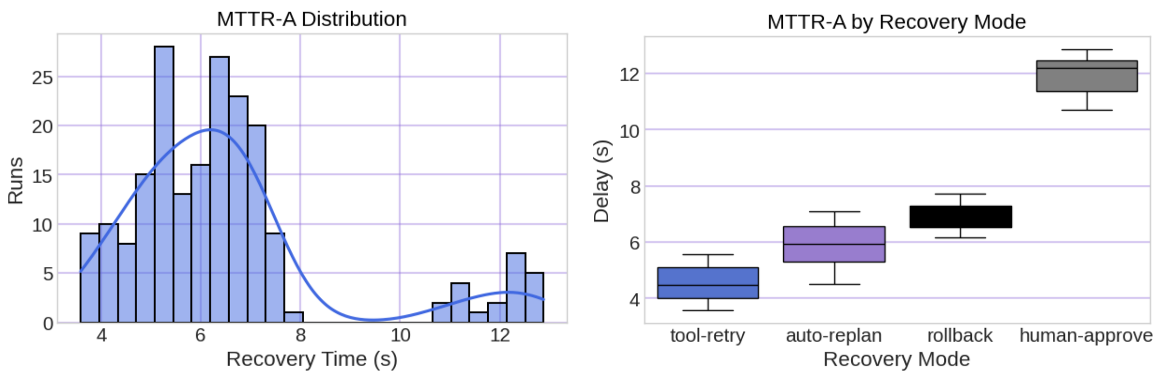

Figure 9: **Distribution and Mode Comparison of** $MTTR\text{-}A_{sys}$**.** Left: overall histogram and kernel density of recovery latency across 200 runs. Right: boxplots per reflex strategy (`tool-retry`, `auto-replan`, `rollback`, `human-approve`).

Temporal Stability Across Runs

Figure 10 visualizes the rolling 20-run MTTR-A average over the 200 iterations. The system exhibits steady recovery performance without trend escalation, suggesting stable reflex control and consistent orchestration efficiency throughout the experiment. Minor oscillations correspond to stochastic differences in recovery mode selection.

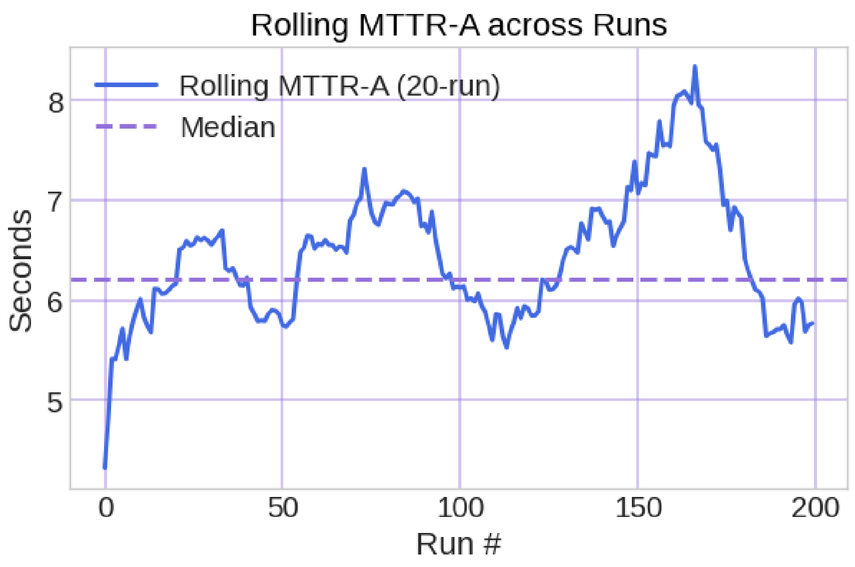

Figure 10: **Rolling MTTR-A across 200 runs.** The blue line represents a 20-run moving average; the dashed red line marks the overall median. No upward drift or performance degradation is observed.

5.1. Latency Composition by Reflex Type

Figure 11 decomposes MTTR-A into three additive components: decomposes MTTR-A into three additive components: detection, decision, and execution latency. Execution time dominates recovery cost across all reflexes, especially in human-approval cases. The low decision latency confirms that policy selection overhead within LangGraph is negligible compared to the actual reflex execution duration. Automated reflexes (auto-replan and tool-retry) recover stability within 5-7 s, maintaining low variance and high throughput. Human oversight introduces expected governance latency but ensures compliance control.

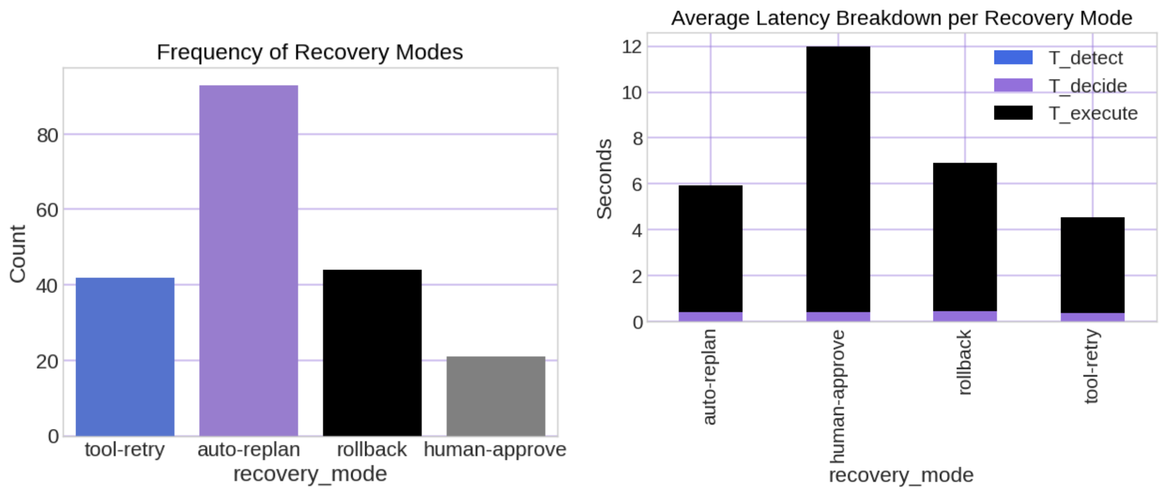

Figure 11: **Latency Composition by Reflex Type.** Left: occurrence frequency of each recovery mode across all runs. Right: mean detection (T_{detect}), decision (T_{decide}), and execution (T_{execute}) times per mode. Execution dominates recovery cost, particularly in human-approval cases.

Comparison to Existing Metrics

In classical dependability research [1], the MTTR quantifies the average duration between a system failure and full service restoration. This formulation assumes binary operational states and deterministic repair procedures. We adapt these classical notions to *agentic cognition*, where recovery refers to reasoning correction, consensus restoration, or human oversight rather than physical service uptime. Traditional MTTR and MTBF describe infrastructure recovery, not cognitive correction. MTTR-A therefore complements-not replaces-these metrics by quantifying recovery latency within multi-agent reasoning loops, bridging classical observability with cognitive control performance.

In LLM-based agents, quality depends on plan consistency, tool correctness, and collaborative coherence. Methods such as ReAct and Tree-of-Thoughts improve *ex-ante* reasoning [14, 16], whereas our approach focuses on *ex-post* reflexes-actions that restore safe state, replan, or escalate after reasoning divergence. From MAS literature, resilient consensus ensures state agreement under faults or attacks, but does not model human-approval gates, sandbox execution, or audit snapshots required in regulated enterprise workflows.

MTTR-A can serve as a building block for higher-level aggregate indicators-such as a *Mean Cognitive Uptime (MCU)* or a *Cognitive Reliability Index (CRI)*-which define measurable targets for adaptive AI performance and governance.

Limitations

The present evaluation relies on a semi-simulated environment with a synthetic corpus and controlled reflex latencies, isolating runtime dependability from semantic variability. Future validation should include open-world orchestration traces (for example, customer-service and data-analysis agents) to test MTTR-A under heterogeneous latency and partial observability. Integrating semantic accuracy metrics such as task-success, BLEU, or F1 alongside MTTR-A would allow joint evaluation of reasoning quality and recovery efficiency, establishing practical trade-offs between cognitive performance and dependability.

6. Conclusion and Future Work

This work adapts classical reliability principles to the cognitive domain of multi-agent systems, proposing a practical framework for evaluating **runtime cognitive dependability**. By extending traditional dependability metrics-MTTR, MTBF, and availability-into the cognitive context, we defined **MTTR-A** as a measurable indicator of reasoning recovery latency and introduced the complementary metrics **MTBF** and **NRR** for continuous assessment of orchestration stability.

The proposed framework integrates a taxonomy of reflexive control actions with empirical evaluation in LangGraph, establishing a reproducible basis for quantifying how distributed agentic systems detect, recover, and stabilize after reasoning drift or coordination faults. This connection between fault-tolerance engineering and cognitive orchestration lays the groundwork for standardized runtime stability benchmarks across LLM-based multi-agent systems.

Future work will extend this foundation to real-world heterogeneous teams, integrating semantic performance metrics with runtime dependability measures to assess trade-offs between reasoning quality and recovery efficiency. Further directions include exploring governance-aware reliability objectives, safety guarantees for reflexive control loops, and alignment with emerging standards for operational AI reliability.

Appendix

A. Proofs

Proof of Theorem 1. Consider the alternating-renewal process $\{(U_n, D_n)\}_{n\geq 1}$ for the MAS, with U_n (cognitive up-time) and D_n (recovery down-time) i.i.d. and $\mathbb{E}[U] = \text{MTBF}_{sys} < \infty$, $\mathbb{E}[D] = \text{MTTR}_{sys} < \infty$. Define $\lambda_{sys} := 1/\text{MTBF}_{sys}$ and $\mu_{sys} := \text{MTTR}_{sys}$.

By the Renewal-Reward Theorem, the steady-state fraction of cognitive up-time is

$$\pi_{\text{up},sys} = \frac{\mathbb{E}[U]}{\mathbb{E}[U] + \mathbb{E}[D]} = \frac{\text{MTBF}_{sys}}{\text{MTBF}_{sys} + \text{MTTR}_{sys}} = \frac{1}{1 + \lambda_{sys}\mu_{sys}}. \tag{A.1}$$

The normalized recovery ratio is defined by

$$\text{NRR}_{sys} := 1 - \lambda_{sys}\mu_{sys}. \tag{A.2}$$

Let $a := \lambda_{sys}\mu_{sys} \geq 0$. Then

$$\frac{1}{1+a} - (1-a) = \frac{a^2}{1+a} \;\geq\; 0, \tag{A.3}$$

with equality iff $a = 0$. Combining (A.1)-(A.3) yields

$$\pi_{\mathrm{up},sys} = \frac{1}{1+\lambda_{sys}\mu_{sys}} \;\geq\; 1 - \lambda_{sys}\mu_{sys} = \mathrm{NRR}_{sys}. \tag{A.4}$$

Moreover, the first-order expansion $(1+a)^{-1} = 1 - a + \mathcal{O}(a^2)$ implies

$$\pi_{\mathrm{up},sys} = \mathrm{NRR}_{sys} + \mathcal{O}\big((\lambda_{sys}\mu_{sys})^2\big)\,, \quad \text{as } \lambda_{sys}\mu_{sys} \to 0. \tag{A.5}$$

This proves that NRR_{sys} is a conservative lower bound on steady-state cognitive up-time and a first-order approximation thereof. $\square$

Proof of Theorem 2. Let R denote the random recovery duration of a single cognitive-fault episode. By Cantelli's one-sided inequality, for any $k > 0$,

$$\Pr[R - \mu \geq k\sigma] \leq \frac{1}{1+k^2}, \tag{A.6}$$

where $\mu = \mathbb{E}[R]$ and $\sigma = \sqrt{\mathrm{Var}[R]}$. Setting $k = k_\alpha = \sqrt{(1-\alpha)/\alpha}$ yields

$$\Pr[R \leq \mu + k_\alpha\sigma] \geq \alpha.$$

Define $R_\alpha = \mu + k_\alpha\sigma$. With probability at least α, the recovery duration satisfies $R \leq R_\alpha$.

The steady-state fraction of cognitive uptime for an alternating-renewal process is

$$\pi_{\mathrm{up},sys} = \frac{1}{1+\lambda_{sys}R}, \tag{A.7}$$

where $\lambda_{sys} = 1/\mathrm{MTBF}_{sys}$ is the fault intensity. Since $(1+x)^{-1}$ is monotonically decreasing for $x > 0$, inequality (A.6) implies

$$\pi_{\mathrm{up},sys} \geq \frac{1}{1+\lambda_{sys}R_\alpha}, \quad \text{with probability at least } \alpha. \tag{A.8}$$

Finally, applying the elementary inequality $(1+x)^{-1} \geq 1 - x$ for $x \geq 0$ gives

$$\pi_{\mathrm{up},sys} \geq 1 - \lambda_{sys}R_\alpha = 1 - \lambda_{sys}(\mu + k_\alpha\sigma) = \mathrm{NRR}_\alpha. \tag{A.9}$$

Thus, with probability at least α, the steady-state cognitive uptime exceeds the confidence-aware lower bound NRR_α, completing the proof. $\square$

B. Query Pool

For completeness, the following query pool was used to induce reasoning drift and test recovery reflexes within the LangGraph experimental workflow. These orchestration-level queries represent typical coordination, governance, and recovery scenarios observed in agentic systems.

```
# === Query Pool ===
QUERY_POOL = [
  "LangGraph recovery reflexes", "agent orchestration reliability",
  "rollback sandbox audit snapshots", "tool retries and backoff",
  "consensus voting disagreement", "policy thresholds and approvals",
  "governance and risk tiers", "observability telemetry signals",
  "drift detection and confidence", "mttra and mtbf calculation",
  "normalized reliability index", "incident playbooks escalation",
  "global memory reconciliation", "safe mode fallback routes",
  "retrieval reranking grounding", "planning decomposition tools"
]
```

Each query was submitted through the reasoning-drift-recovery cycle defined in Section 4, ensuring consistent cognitive fault induction and enabling reproducible measurement of recovery latency and MTTR-A dynamics.

C. Experimental Configuration

All experiments were executed on Google Colab with an NVIDIA A100 GPU, using Python 3.10 and the LangGraph 0.0.52 library. Each run instantiated a stateful graph of three nodes (*Reasoning*, *Drift Check*, and *Recovery*), executed 200 independent iterations with random seeds fixed to ensure reproducibility.

References

[1] Betsy Beyer, Jennifer Jones, Jennifer Petoff, and Niall Murphy. *Site Reliability Engineering: How Google Runs Production Systems.* O'Reilly Media, 2020.

[2] S. Zhou et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

[3] D. Amodei et al. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[4] S. Han et al. Llm multi-agent systems: Challenges and open problems. *arXiv preprint arXiv:2402.03578*, 2025.

[5] T. Guo et al. Large language model-based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.

[6] Z. Sun et al. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*, 2024.

[7] L. Wang et al. A survey on llm-based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023.

[8] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[9] Google DeepMind. Gemini 1.0 technical report. *arXiv preprint arXiv:2312.11805*, 2023.

[10] Y. Bai, A. Kadavath, S. Kundu, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[11] M. Varga and K. B. Loo. Modelling resilient collaborative multi-agent systems. *Computing (Springer)*, 2020.

[12] R. E. Barlow and F. Proschan. *Mathematical Theory of Reliability.* SIAM, 1996. Original edition 1965.

[13] J. Wei et al. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS 2022 / arXiv:2201.11903*, 2022.

[14] S. Yao et al. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[15] T. Schick et al. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[16] S. Yao et al. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[17] J. S. Park et al. Generative agents: Interactive simulacra of human behavior. *ACM UIST / arXiv:2304.03442*, 2023.

[18] G. Li et al. Camel: Communicative agents for 'mind' exploration of llm society. *NeurIPS 2023 / arXiv:2303.17760*, 2023.

[19] N. Shinn et al. Reflexion: Language agents with verbal reinforcement learning. *NeurIPS 2023 / OpenReview*, 2023.

[20] G. Wang et al. Voyager: An open-ended embodied agent with llms. *arXiv preprint arXiv:2305.16291*, 2023.

[21] Z. Xi et al. The rise and potential of llm-based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

[22] X. Bo, Z. Zhang, Q. Dai, X. Feng, L. Wang, R. Li, and J. R. Wen. Reflective multi-agent collaboration based on large language models. *NeurIPS 2024*, 2024.

[23] Y. Zhang, R. Sun, Y. Chen, T. Pfister, R. Zhang, and S. Arik. Chain of agents: Large language models collaborating on long-context tasks. *NeurIPS 2024*, 2024.

[24] W. Tao, Y. Zhou, Y. Wang, W. Zhang, H. Zhang, and Y. Cheng. Magis: Llm-based multi-agent framework for github issue resolution. *NeurIPS 2024*, 2024.

[25] A. Estornell and Y. Liu. Multi-llm debate: Framework, principles, and interventions. *NeurIPS 2024*, 2024.

[26] H. Bai, Y. Zhou, J. Pan, M. Cemri, A. Suhr, S. Levine, and A. Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *NeurIPS 2024*, 2024.

[27] W. Li, W. E. Bishop, A. Li, C. Rawles, F. Campbell-Ajala, D. Tyamagundlu, and O. Riva. On the effects of data scale on ui control agents. *NeurIPS 2024 / arXiv:2406.03679*, 2024.

[28] J. Liu et al. Secure consensus tracking under asynchronous dos. *International Journal of Control, Automation and Systems*, 2023.

[29] M. Wang et al. Resilient consensus control for linear mas under false data injection. *International Journal of Control, Automation and Systems*, 2023.

[30] C. Gao et al. Fault-tolerant consensus control of mas: A survey. *International Journal of Systems Science*, 2022.

[31] C. Liu et al. Fault-tolerant consensus for mas with actuator/sensor faults. *Information Sciences*, 2023.

[32] J. Shi et al. Robust cooperative fault-tolerant control for uncertain mas. *Sensors*, 2024.

[33] L. Sheng et al. Fault-tolerant formation consensus for time-varying mas. *International Journal of Network Dynamics & Intelligence*, 2024.

[34] H. Lin, T. Zhou, and M. Xu. Alas: A stateful multi-llm framework with runtime monitoring and scheduling. *arXiv preprint arXiv:2505.12501*, 2025.

[35] R. Singh, J. Chen, and X. Wang. Advancing agentic systems: Dynamic task-graph decomposition and runtime metrics. *arXiv preprint arXiv:2410.22457*, 2024.

[36] R. Cao, F. Lei, H. Wu, J. Chen, Y. Fu, H. Gao, and T. Yu. Spider2-v: How far are multimodal agents from automating data science and engineering workflows? *NeurIPS 2024*, 2024.