Rillor

RESEARCH REPORT

Why multi-agent LLM systems fail: what the failure studies show

The MAST taxonomy, debate studies and failure attribution benchmarks show why multi-agent LLM systems fail: loose specifications, dropped correct answers, shallow verification and errors that spread.



Rillor

Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

Failure studies of multi-agent LLM systems published from 2024 to 2026 trace many failures to how agents are specified, how they share information and how their work is checked [1, 7, 11].

- The Multi-Agent System Failure Taxonomy (MAST), from the NeurIPS 2025 Datasets and Benchmarks Track paper "Why Do Multi-Agent LLM Systems Fail?", sorts the failures of multi-agent LLM systems into 14 modes in three categories: system design issues, inter-agent misalignment and task verification [1, 2]. It rests on 1,642 annotated execution traces from seven open-source frameworks run on benchmark tasks [1].
- In MAST's traces, step repetition (15.7% of observed failures), a mismatch between an agent's reasoning and its action (13.2%) and unawareness of when to stop (12.4%) lead the 14 modes [1]. Missing or incorrect verification also appears often in runs that succeed [1].
- MAST measured failure rates of 41% to 86.7% across seven multi-agent systems on different benchmark tasks, so its authors call the rates not directly comparable [1]. No source reviewed for this report measures a production failure rate for multi-agent systems.
- In an October 2026 preprint on 3,211 failed multi-agent discussions, failures to recover from or correct an error appeared in 94.9% and failures in reaching a group decision in 83.4%, and correct minority answers were often suppressed, more so in larger groups [7]. A separate study found that majority voting alone accounts for most of the gains attributed to multi-agent debate [9].
- Errors spread further in some team designs. Agents working with no coordination amplified errors counted in execution traces 17.2 times, against 4.4 times under a central orchestrator that checks work before combining it, though error amplification did not significantly predict performance once other coordination measures were controlled [11]. In star-shaped teams, an error injected at the hub reached an infection rate of 100.0% in both frameworks tested, against 15.9% and 9.7% when injected at a leaf [12].
- Finding the failing agent and step is hard. On the Who&When benchmark the top method identified the responsible agent 53.5% of the time and the decisive step 14.2% of the time [16]. On TraceElephant, a 2026 benchmark of 220 failure traces, full execution traces raised attribution accuracy to 65.9% for agents and 30.3% for steps, gains of 22% and 76% over traces that keep only agent outputs [17].
- Measured fixes helped in part. MAST's prompt and topology changes raised one system from 25.0% to 34.4% and 40.6% task accuracy on 32 tasks, which its authors judged not substantial [1]. Retrieval cut discussion failures from 74.6% to 36.4% [7], and an inspector agent combined with a challenger recovered 96.4% of the performance faulty agents had cost one system [13].

Background

What a multi-agent LLM system is

The MAST paper defines an LLM-based agent as "an artificial entity with prompt specifications (initial state), conversation trace (state), and ability to interact with the environments such as tool usage (action)" [1]. A multi-agent system is "a collection of agents designed to interact through orchestration, enabling collective intelligence," and failures are "instances where the MAS does not achieve its intended task objectives" [1].

The systems studied differ in how agents connect. In a star topology a supervisor directs service agents, in a hierarchy a planner directs child agents, and in an assembly line each role hands its work to the next [1]. Kim et al. compare a single agent with four multi-agent designs: independent agents with no coordination mechanism, a centralized orchestrator, decentralized peers and a hybrid [11].

Why the question arose

MAST's authors began from the observation that the performance gains of multi-agent systems on popular benchmarks "are often minimal" [1]. Practitioner reports describe the same trouble. A model developer reported in June 2025 that its early research agents made errors "like spawning 50 subagents for simple queries," and that its multi-agent system used about 15 times the tokens of a chat [30]. In January 2026 the same developer put multi-agent token use at 3 to 10 times that of single-agent approaches for equivalent tasks [31].

Hammond et al. frame multi-agent risk around incentives, with three failure modes: miscoordination, conflict and collusion [28]. Miscoordination "arises when agents, despite a mutual and clear objective, cannot align their behaviours to achieve this objective" [28]. This report covers teams built for one shared task and, in Finding 8, outside content that turns such a team against its task.

When to split a task across agents at all is a separate question, covered in when a multi-agent system beats a single agent (https://rillor.com/insights/single-agent-vs-multi-agent-systems). This report stays on how and why split systems fail.

Method

What was examined. Published evidence on why multi-agent LLM systems fail: which failures the studies observe, how often, where in a run they start and spread, how well they can be traced to an agent and a step, and which fixes were measured.

Sources. The MAST paper and its records [1, 2, 3, 4, 5, 6], communication and debate studies [7, 8, 9, 10], propagation and topology studies [11, 12, 13, 14, 15], failure attribution benchmarks [16, 17, 18, 19, 20, 21, 22], studies and standards on adversarial content [23, 24, 25, 26, 27, 29], a risk framework [28], practitioner reports [30, 31, 32, 33] and a preprint that restates MAST's range [34]. Every source was viewed on 7 November 2026.

How. Each figure was taken from the section, table or caption cited and stays with its benchmark, version and date. Figures from different studies are never pooled. Where MAST's versions differ, the October 2025 version is used, and the older figures appear only in Finding 10. That version gives shares per failure mode only, so the category totals here are sums of its mode figures, marked "derived". MAST's intervention gains are percentage points on small task sets and are reported as such.

Two corpora were labelled mostly by LLM annotators calibrated on human samples, MAST-Data [1] and the communication-failure taxonomy [7], and their agreement figures are given with them. One system's error codes were discovered and applied by an LLM with no reported human check [15]. One attribution dataset, Aegis, holds 9,533 trajectories made by "a LLM-based manipulator" that injects errors into successful runs [22], so its errors are synthetic. It informs the method discussion and supplies no failure rate. Models and model vendors are named only in Sources. Studied frameworks are named where a finding is specific to one.

Findings

1. MAST: 14 failure modes in three categories

MAST was built with Grounded Theory from "over 150 MAS execution traces (each averaging over 15,000 lines of text)" from five frameworks, by six expert annotators [1]. Three annotators labelled 15 traces "until achieving high inter-annotator agreement (κ = 0.88)" [1]. An LLM annotator built on a reasoning model was then calibrated to κ = 0.77 against the experts, and a further round of human annotation on two systems and benchmarks outside the initial study reached κ = 0.79 [1].

The resulting dataset, MAST-Data, holds 1,642 annotated traces from seven frameworks, run mainly with two model families, plus two open-weight models on one benchmark, on coding, math and general agent tasks [1]. Its dataset card lists a CC BY 4.0 license [6].

Code	Failure mode	Share of observed failures
FM-1.1	Disobey task specification	11.8%
FM-1.2	Disobey role specification	1.5%
FM-1.3	Step repetition	15.7%
FM-1.4	Loss of conversation history	2.80%
FM-1.5	Unaware of termination conditions	12.4%
FM-2.1	Conversation reset	2.20%
FM-2.2	Fail to ask for clarification	6.80%
FM-2.3	Task derailment	7.40%
FM-2.4	Information withholding	0.85%
FM-2.5	Ignored other agent's input	1.90%
FM-2.6	Reasoning-action mismatch	13.2%
FM-3.1	Premature termination	6.20%

Code	Failure mode	Share of observed failures
FM-3.2	No or incomplete verification	8.20%
FM-3.3	Incorrect verification	9.10%

Source: [1], Section 4. Summed by category (derived), system design issues come to 44.2%, inter-agent misalignment to 32.35% and task verification to 23.5%, a total of 100.05% from rounding [1].

The categories overlap little. Correlations between them are "low (0.17-0.32)", while some modes show "moderate correlations (max of 0.63)" that "might lead automated evaluators to conflate distinct root causes" [1]. The authors add: "we do not claim it covers every potential failure pattern" [1].

Across seven systems, MAST reports failure rates from 41% to 86.7% [1]. The caption of the figure behind that range says the performances "are measured on different benchmarks, therefore they are not directly comparable" [1]. ProgramDev, the authors' own benchmark of 30 tasks "such as Tic-Tac-Toe, Chess, or Sudoku", was designed "relatively straightforward for MAS" so as "to better isolate specific failure dynamics" [1].

2. System design issues: repetition, specifications and stopping

Step repetition, the "unnecessary reiteration of previously completed steps in a process", accounts for 15.7% of observed failures [1]. Unawareness of termination conditions accounts for 12.4% and disobeying the task specification for 11.8% [1]. In one example, a framework asked for a Wordle game without a fixed word bank "still produces code with a fixed list and new errors" [1].

The authors draw two conclusions about design. "MAS failure is not merely a function of challenges in the underlying model," and "a well-designed MAS can result in performance gain when using the same underlying model" [1]. They also "conjecture that improvements in the base model capabilities" fall short of addressing every mode, and that many failures "arise from the challenges in organizational design and agent coordination" [1].

Each system fails in its own way. One system with a star topology "frequently suffers from premature terminations", and another "exhibits a tendency towards step repetition" [1]. The authors conclude that "there is no one-size-fits-all solution to MAS failures" [1]. Harder benchmarks brought more failures: across GSM, MMLU and OlympiadBench in one framework, "the more challenging the benchmark is ... the higher the failure rate" [1].

Practitioners report the same patterns. A model developer wrote in June 2025: "Without detailed task descriptions, agents duplicate work, leave gaps, or fail to find necessary information" [30]. In an error analysis of the generalist system Magentic-One, "the most common code, persistent-inefficient-actions," covers agents that "repeatedly engage in unproductive behaviors without adapting their strategies" [15].

3. Inter-agent misalignment and the communication-failure taxonomy

Reasoning-action mismatch, a "discrepancy between the logical reasoning process and the actual actions taken by the agent", ranks second among MAST's modes at 13.2% [1]. Information withholding is rare at 0.85%, defined as a "failure to share or communicate important data or insights" that could change other agents' decisions [1].

MAST's authors find that standard message protocols do not remove these failures. "Solutions focused on context or communication protocols are often insufficient for FC2 failures, which demand deeper 'social reasoning' abilities from agents" [1]. Naming the Model Context Protocol and Agent to Agent, they note that "the errors we observe in FC2 occur even when agents within the same framework communicate using natural language," and call this "the collapse of 'theory of mind'" [1].

Pitre et al. built a second taxonomy, focused on communication, from 3,211 failed multi-round trajectories across medical questions, coding, finance, multi-hop reasoning, ethics and embodied text tasks [7]. Its agents came from open-weight models of 3B to 120B parameters, and the taxonomy has six parent categories with "24 child labels and 11 final leaf labels" [7]. In validation, 93.4% of human-identified errors mapped to a fine-grained label, annotators agreed at 0.82 (Jaccard) on parent labels and automatic labels scored 0.91 and 0.84 micro-F1 [7].

Two fine-grained labels stood out: "persistent wrong framing after correction and premature consensus, which appear in 62.1% and 59.6% of failed trajectories" [7]. Failures of recovery and correction overlap failures of consensus and decision "in 78.6% of failed trajectories" [7].

Practitioners describe the same loss of information at handoffs. A model developer wrote in January 2026 that role-split agents "engage in a 'telephone game,' passing information back and forth with each handoff degrading fidelity" [31]. An agent-software company wrote in June 2025 that "actions carry implicit decisions, and conflicting decisions carry bad results" [32]. In April 2026 it added that "parallel agents make implicit choices about style, edge cases, and code patterns" [33].

4. Verification: shallow checks and early stops

MAST's verification category holds three modes: premature termination (6.2%), no or incomplete verification (8.2%) and incorrect verification (9.1%) [1].

The authors conclude that "multi-level verification is needed" and that "sole reliance on final-stage, low-level checks is inadequate" [1]. Systems with explicit verifiers "generally show fewer total failures", yet "the presence of a verifier is not a silver bullet" [1]. In one case a chess program "passes superficial checks (e.g., code compilation) but contains runtime bugs" [1]. Many verifiers "perform only superficial checks, despite being prompted to perform thorough verification, such as checking if the code compiles or if there are leftover TODO comments" [1].

Design choices trade one category for another. With the same model on the same benchmark, the assembly-line system MetaGPT showed "60-68% less failure in FC1 and FC2" than the hierarchical ChatDev, and "1.56x more FC3 failure" [1].

Successful runs are "not failure-free" [1]. Unawareness of termination conditions and information withholding "appear almost exclusively in failed runs", while incorrect and incomplete verification "appear frequently even in successful runs" [1]. In ChatDev, incorrect verification appeared in 20.0% of successful runs and 25.0% of failed ones [1].

Other sources report the same gap. Magentic-One's analysis named insufficient verification steps "the second-most common code", covering tasks "marked as complete without thorough validation" [15]. A model developer wrote in January 2026 that "the most significant failure mode for verification subagents is marking outputs as passing without thorough testing" [31]. In a two-week red-teaming study of agents in a live laboratory environment, "agents reported task completion while the underlying system state contradicted those reports" in several cases [29].

5. Debate and conformity

Wynn, Satija and Hadfield found that debate "can lead to a decrease in accuracy over time", even in settings where more capable models outnumber weaker ones [8]. On CommonSenseQA, "debate always harms performance" [8]. The share of correct-to-incorrect transitions exceeded the share of incorrect-to-correct ones in later rounds, undesirable flips were likelier "when the ego agent is isolated", and a prompt that rewarded correctness "does not significantly reduce the likelihood" of such flips [8].

Choi, Zhu and Li found across seven benchmarks that "Majority Voting alone accounts for most of the performance gains typically attributed to" multi-agent debate [9]. They show that debate "induces a martingale over agents' belief trajectories, implying that debate alone does not improve expected correctness" [9]. An earlier ICML 2024 study found that debating systems "do not reliably outperform other proposed prompting strategies, such as self-consistency and ensembling", and are "more sensitive to different hyperparameter settings and difficult to optimize" [10].

Pitre et al. locate when the trouble starts. "Round 2 is a critical point where many failures emerge after agents first observe peer responses, and correct minority signals are often suppressed rather than amplified, especially in larger groups" [7]. Between three and seven agents, individual rescue fell from 7.87% to 2.50% and individual suppression rose from 11.37% to 17.70% [7]. Incorrect traces still ended "near 90% confidence" [7].

Mistakes alone do not separate success from failure. "Even among 500 successful trajectories, 91.2% contain at least one error" [7]. Prevalence in successful against failed trajectories was 0.37 against 0.94 for recovery failures, 0.51 against 0.93 for consensus failures and 0.08 against 0.60 for persistent wrong framing, while reasoning errors (0.35 against 0.35) and tool and evidence errors (0.39 against 0.38) were about equal [7].

6. Error propagation and topology

Kim et al. evaluated "260 configurations spanning six agentic benchmarks, five canonical architectures" and "three LLM families", with tools, prompts and compute standardized [11]. Against a single agent, results ranged from "+80.8% on decomposable financial reasoning to -70.0% on sequential planning" [11].

Architecture	Trace-level error amplification	Communication overhead	Coordination failure share
Single agent	1.0	0%	Does not apply
Independent	17.2	58%	0%
Centralized	4.4	285%	1.8%
Decentralized	7.8	263%	3.2%
Hybrid	5.1	515%	12.4%

Source: [11], Sections 1, 3 and 4.4 and the appendix error taxonomy.

The authors write that "Independent systems amplify trace-level errors 17.2× through unchecked error propagation," while centralized coordination "contains this to 4.4× by enforcing validation bottlenecks that intercept errors before aggregation" [11]. The amplification factor is "estimated from execution-trace token analysis", and the task-level factor is about 1.1 to 1.3, against about 4 to 17 at the trace level [11]. After controls, "neither the main effect of error amplification ($\beta = 0.014$, $p = 0.658$) nor its interaction with tool count" reaches statistical significance [11]. Context omission ran at 15.8% to 25.2% at baseline, fell to 8.3% under a centralized orchestrator and stayed at 24.1% with independent agents [11].

Xie et al. ran six frameworks in chain, mesh and star topologies and describe "cascade amplification, topological sensitivity, and consensus inertia" [12]. Collaboration "may cause minor inaccuracies to gradually solidify into system-level false consensus through iteration" [12]. An error injected at the hub reached an infection rate of 100.0% in both star frameworks, against 15.9% and 9.7% when injected at a leaf, with impact factors of 6.29 and 10.31 [12]. They found a falsehood cheap to introduce and its correction more costly as the workflow progresses [12].

Huang et al. compared structures with faulty agents across four tasks and six systems [13]. The hierarchical structure "exhibits superior resilience with the lowest performance drop of 5.5%, compared to 10.5% and 23.7% of other two structures" [13].

Ma et al. analyzed "307 real-world failure cases collected from two representative agentic workflow platforms", across ten workflows [14]. "In 32% of failures, the root cause node differs from the failure symptom node, with propagation distances exceeding 40% of the workflow length in over 10% of cases" [14].

7. Finding the agent and step that failed

Who&When, published at ICML 2025, holds "failure logs from 127 LLM multi-agent systems" and 184 failure annotation tasks [16]. "The best method achieves 53.5% accuracy in identifying failure-responsible agents but only 14.2% in pinpointing failure steps, with some methods performing below random" [16]. On the hand-crafted systems, the top method reached "only 8.77% accuracy in identifying decisive error steps" [16]. Labelling took the three annotators 30.9, 30.2 and 23.2 hours, and uncertain labels ranged "from 15% to 30%" across annotators [16].

Later benchmarks change what attribution methods can see. TraceElephant, accepted at ACL 2026, argues that earlier benchmarks "rely on partially observable traces that capture only agent outputs" [17]. On 220 failure traces from three systems, full traces gave "an average attribution accuracy of 65.9% at the agent level and 30.3% at the step level", gains of 22% and 76% over output-only traces [17].

MP-Bench, a March 2026 preprint, finds that "MAS failures often admit multiple plausible attributions" [18]. Over 289 logs and about 346 expert-hours, "only 16.2% of steps are annotated as failure-inducing by all three annotators", 27.8% by exactly two and 56.1% by one [18]. Its authors argue that earlier conclusions that LLMs struggle with attribution "are largely driven by limitations in existing benchmark designs" [18].

Benchmark	Data	Reported result
Who&When (ICML 2025)	Logs from 127 systems, 184 tasks	53.5% agent, 14.2% step [16]
TRAIL (2025)	148 human-annotated traces, single and multi-agent	Top model tested scored 11% [21]
AgentRx (EMNLP Findings 2026)	170 trajectories, 11 task settings, $\kappa = 0.89$	Step localization up 75% on average over prior work [19]
TraceElephant (ACL 2026)	220 failure traces, 3 systems	65.9% agent, 30.3% step with full traces [17]
MP-Bench (preprint, 2026)	289 logs, three experts	16.2% of failure steps marked by all three [18]

Benchmark	Data	Reported result
DuoTrace (preprint, 2026)	Who&When, six attribution methods	Detect-before-attribute filter, average absolute gains of 11.2 points (agent) and 7.7 points (step) [20]

DuoTrace finds that LLM-based attribution methods "suffer from long-context degradation over lengthy trajectories" [20]. A model developer noted in June 2025 that agents "are non-deterministic between runs, even with identical prompts" and that "this makes debugging harder" [30]. An agent-software company advised in June 2025: "Share context, and share full agent traces, not just individual messages" [32].

8. Adversarial content that spreads between agents

Triedman, Jha and Shmatikov found that "adversarial content can hijack control and communication within the system to invoke unsafe agents and functionalities" [23]. With the agents running on one model, web-based attacks made the system "execute arbitrary malicious code in 58-90% of trials (depending on the orchestrator)", and "in some model-orchestrator configurations, the attack success rate is 100%" [23]. The attacks "succeed even if individual agents are not susceptible to direct or indirect prompt injection, and even if they refuse to perform harmful actions" [23].

Lee and Tiwari describe prompts that "self-replicate across interconnected agents, behaving much like a computer virus" [24]. Self-replicating success was "about 20% lower in local messaging compared to global messaging" [24]. Neither LLM Tagging nor traditional defenses alone were sufficient, while "when combined, they provide robust protection" [24]. Xie et al. found attack success from plain injection "near zero in almost all settings", while packaging the falsehood to match an agent's role raised it mostly to 85% to 100% in all but one framework [12].

Two industry documents name these risks. The OWASP Top 10 for Agentic Applications for 2026 lists ASI07, insecure inter-agent communication, which "occurs when these exchanges lack proper authentication, integrity, or semantic validation" [25]. ASI08, cascading failures, covers "a single fault (hallucination, malicious input, corrupted tool, or poisoned memory)" that "propagates across autonomous agents, compounding into system-wide harm", and "focuses on the propagation and amplification of faults rather than their origin" [25]. A cloud provider's red team counts among novel failures those "that occur in the communication flow between agents within a multiagent system" [27], and lists agent injection, agent impersonation, agent flow manipulation and multi-agent jailbreaks [26].

9. What the measured fixes did

MAST's authors tested two interventions [1].

System, model and benchmark	Baseline	Improved prompts	New topology
AG2 MathChat, older model, 200 GSM-Plus problems	84.75 ± 1.94	89.75 ± 1.44	85.50 ± 1.18
AG2 MathChat, newer model of the same family, 200 GSM-Plus problems	84.25 ± 1.86	89.00 ± 1.38	88.83 ± 1.51
ChatDev, ProgramDev-v0 (32 tasks)	25.0	34.4	40.6
ChatDev, HumanEval	89.6	90.3	91.5

Source: [1], Appendix H, Table 5. Accuracy in percent, six repetitions for the math runs.

In the AG2 MathChat runs, the topology gain with the older model was not significant (p = 0.4), and with the newer model both changes reached p = 0.03 [1]. For ChatDev the prompt change enforced a hierarchy, so that "only superior agents can finalize conversations", and the topology change moved "from a directed acyclic graph (DAG) to a cyclic graph" that ends only when a lead agent confirms all reviews, "with a maximum iteration cutoff" [1]. The paper's "+9.4%" and "+15.6%" are percentage points on 32 tasks (derived: 34.4 minus 25.0 and 40.6 minus 25.0) [1]. The authors write that "these strategies are not universal" and that "they do not constitute substantial improvements" [1].

Evidence access moved more. In Pitre et al., "retrieval reduces the overall failure rate from 74.6% to 36.4% on HotpotQA and MuSiQue", while "increasing the team from three to seven agents does not significantly reduce failures" [7]. "Hierarchy reduces oscillation but not overall failures, while explicit roles have no detectable effect" [7]. The authors recommend "stronger disagreement handling, evidence verification, minority-signal preservation, and recovery checks before finalization" [7].

Checks on messages also helped. Huang et al.'s Inspector, "an additional agent to review and correct messages", combined with their Challenger, recovered "96.4% of the performance loss on the Self-collab system" [13]. Xie et al.'s message-layer governance plug-in kept final outputs uninfected in 89% to 94% of runs on average across its three modes, with a low of 58.3% for one framework and attack, while containment with reflection fell to 5.0% and 19.2% on one framework [12]. In Magentic-One's ablation, "without the full ledgers, performance drops by 31%" [15].

Practitioners narrowed where they split work. A model developer advised in January 2026 that "work should only be split when context can be truly isolated" [31]. An agent-software company wrote in April 2026 that what works is "setups where multiple agents contribute intelligence to a task while writes stay single-threaded", and that "clean context leads to a notable improvement in capabilities when using a generator-verifier loop" [33].

10. Where the sources disagree

MAST's versions. The April 2025 version called the dataset MAD and gave category shares of 41.77%, 36.94% and 21.30% [4]. The conference site still shows an older abstract with "MAD, a comprehensive dataset of 1000+ annotated traces" and "specification issues" as the first category [3]. The October 2025 version says MAST-Data with 1,642 traces, and the proceedings abstract "1600+ annotated traces" [1, 2]. The project's code repository still links the April version and the MAD dataset [5]. The April version also disagrees with itself on one system, at "33.33% correctness" on ProgramDev in one place (66.67% failure, derived) and "75% failure" in another [4].

Trace level and task level. The 17.2 and 4.4 amplification factors are trace-level estimates [11]. At the task level the factors are about 1.1 to 1.3, and error amplification does not significantly predict performance once other coordination measures are controlled [11].

How hard attribution is. Who&When reports low accuracy and some methods below random [16]. TraceElephant shows that full traces raise accuracy [17], and MP-Bench argues that low scores largely reflect benchmark design [18].

Benchmark against production. A May 2026 preprint opens with "Multi-agent LLM systems fail in production at rates between 41% and 87%" [34]. MAST measured that range on different benchmarks and calls the rates not directly comparable [1].

Model generations. MAST conjectures that better base models fall short of addressing every failure mode [1]. In April 2026 an agent-software company wrote that models had become more naturally agentic [33]. No study reviewed reruns MAST's taxonomy on 2026 models.

Implications

For teams building multi-agent systems. The studies point to where checks belong. Validation before results are combined contained trace-level error amplification in one controlled study [11], errors at a hub spread further than errors at a leaf [12], and 32% of workflow failures surfaced away from their cause [14]. Verifiers that check only whether code compiles missed runtime bugs [1], while checks against evidence and on messages moved the numbers: retrieval cut discussion failures from 74.6% to 36.4% [7] and an inspector paired with a challenger recovered 96.4% of the performance faulty agents had cost one system [13]. Independent answers need protection, because correct minority answers were suppressed more in larger groups [7] and majority voting over agents' answers accounted for most debate gains [9]. Outside content can steer a whole team even when each agent resists injection alone [23], so inter-agent messages need the authentication, integrity and semantic validation whose absence ASI07 describes [25].

For teams evaluating or buying them. A failure rate means little without its benchmark, version and date [1, 34]. Successful runs contained errors in 91.2% of one sample [7] and verification failures in MAST's traces [1], so a pass on the final answer can hide problems in the run. Runs differ even with identical prompts [30], so one run says little about reliability. Ask whether a system logs full traces, since output-only traces lowered attribution accuracy [17].

Every Rillor agent system has the same parts: a planner and specialist agents in the orchestrator-worker pattern, with an audit step before any answer, and every step is logged so a run can be traced and replayed.

Limits of this analysis

- Every failure rate reviewed comes from benchmark tasks or lab setups. No source measures a production failure rate for multi-agent systems.
- MAST-Data and the communication-failure taxonomy were labelled mostly by LLM annotators calibrated on human samples [1, 7], and one system's error codes were made by an LLM with no reported human check [15].
- MAST's traces come from two main model families, plus 400 traces from two open-weight models [1]. No study reviewed reruns its taxonomy on newer models.
- Many sources are preprints. Pitre et al. is a preprint posted on 23 October 2026 [7].
- Several papers carry many results in figures, which were read here from text, captions and alt text.
- Practitioner posts describe their authors' own systems and are vendor-reported [30, 31, 32, 33].
- Aegis's errors are synthetic [22], and the studies of adversarial content describe attacks in research setups [23, 24].

Sources

1. Cemri, M., Pan, M. Z., Yang, S. et al. Why Do Multi-Agent LLM Systems Fail? arXiv 2503.13657 v3. arXiv, 26 October 2025, viewed 7 November 2026. https://arxiv.org/abs/2503.13657

2. Cemri, M. et al. Why Do Multi-Agent LLM Systems Fail? Advances in Neural Information Processing Systems 38 (NeurIPS 2025), Datasets and Benchmarks Track. NeurIPS Proceedings, 2025, viewed 7 November 2026. https://papers.nips.cc/paper_files/paper/2025/hash/b1041e52d3be19f0a9bc491657488e4a-Abstract-Datasets_and_Benchmarks_Track.html

3. Why Do Multi-Agent LLM Systems Fail? Poster, Workshop: Evaluating the Evolving LLM Lifecycle. NeurIPS 2025 conference site, 2025, viewed 7 November 2026. https://neurips.cc/virtual/2025/loc/san-diego/122442

4. Cemri, M. et al. Why Do Multi-Agent LLM Systems Fail? arXiv 2503.13657 v2. arXiv, 22 April 2025, viewed 7 November 2026. https://arxiv.org/html/2503.13657v2

5. MAST, project repository. GitHub, last push 23 July 2025, viewed 7 November 2026. https://github.com/multi-agent-systems-failure-taxonomy/MAST

6. mcemri/MAST-Data, dataset card. Hugging Face, last modified 13 August 2026, viewed 7 November 2026. https://huggingface.co/datasets/mcemri/MAST-Data

7. Pitre, P., Hou, J., Fang, Y., Ramakrishnan, N. and Wang, X. Understanding Communication Failures in Multi-Agent LLM Systems. arXiv 2610.32060 v1. arXiv, 23 October 2026, viewed 7 November 2026. https://arxiv.org/abs/2610.32060

8. Wynn, Satija and Hadfield. Talk Isn't Always Cheap: Understanding Failure Modes in Multi-Agent Debate. arXiv 2509.05396 v2. arXiv, 13 October 2025, viewed 7 November 2026. https://arxiv.org/abs/2509.05396

9. Choi, Zhu and Li. Debate or Vote: Which Yields Better Decisions in Multi-Agent Large Language Models? arXiv 2508.17536 v2. arXiv, 23 October 2025, viewed 7 November 2026. https://arxiv.org/abs/2508.17536

10. Smit et al. Should we be going MAD? A Look at Multi-Agent Debate Strategies for LLMs. Proceedings of the 41st International Conference on Machine Learning, PMLR 235. PMLR, 2024, viewed 7 November 2026. https://proceedings.mlr.press/v235/smit24a.html

11. Kim et al. Towards a Science of Scaling Agent Systems. arXiv 2512.08296 v3. arXiv, 8 April 2026, viewed 7 November 2026. https://arxiv.org/abs/2512.08296

12. Xie et al. From Spark to Fire: Modeling and Mitigating Error Cascades in LLM-Based Multi-Agent Collaboration. arXiv 2603.04474 v2. arXiv, 11 May 2026, viewed 7 November 2026. https://arxiv.org/abs/2603.04474

13. Huang et al. On the Resilience of LLM-Based Multi-Agent Collaboration with Faulty Agents. arXiv 2408.00989 v4. arXiv, 28 May 2025, viewed 7 November 2026. https://arxiv.org/abs/2408.00989

14. Ma et al. Demystifying the Lifecycle of Failures in Platform-Orchestrated Agentic Workflows. arXiv 2509.23735 v2. arXiv, 27 February 2026, viewed 7 November 2026. https://arxiv.org/abs/2509.23735

15. Fourney et al. Magentic-One: A Generalist Multi-Agent System for Solving Complex Tasks. arXiv 2411.04468 v1. Microsoft Research, 7 November 2024, viewed 7 November 2026. https://arxiv.org/abs/2411.04468

16. Zhang et al. Which Agent Causes Task Failures and When? On Automated Failure Attribution of LLM Multi-Agent Systems. Proceedings of the 42nd International Conference on Machine Learning, PMLR 267. PMLR, 2025, viewed 7 November 2026. https://proceedings.mlr.press/v267/zhang25cq.html

17. Chen et al. Seeing the Whole Elephant: A Benchmark for Failure Attribution in LLM-based Multi-Agent Systems. arXiv 2604.22708 v1. arXiv, 24 April 2026, viewed 7 November 2026. https://arxiv.org/abs/2604.22708

18. In et al. Rethinking Failure Attribution in Multi-Agent Systems: A Multi-Perspective Benchmark and Evaluation. arXiv 2603.25001 v1. arXiv, 26 March 2026, viewed 7 November 2026. https://arxiv.org/abs/2603.25001

19. Barke et al. AgentRx: Diagnosing AI Agent Failures from Execution Trajectories. arXiv 2602.02475 v2. arXiv, 31 August 2026, viewed 7 November 2026. https://arxiv.org/abs/2602.02475

20. Zhang et al. Cascade Failure Attribution for Multi-Agent Systems. arXiv 2608.29646 v2. arXiv, 19 October 2026, viewed 7 November 2026. https://arxiv.org/abs/2608.29646

21. Deshpande et al. TRAIL: Trace Reasoning and Agentic Issue Localization. arXiv 2505.08638 v3. arXiv, 23 June 2025, viewed 7 November 2026. https://arxiv.org/abs/2505.08638

22. Kong et al. Aegis: Automated Error Generation and Attribution for Multi-Agent Systems. arXiv 2509.14295 v6. arXiv, 27 April 2026, viewed 7 November 2026. https://arxiv.org/abs/2509.14295

23. Triedman, Jha and Shmatikov. Multi-Agent Systems Execute Arbitrary Malicious Code. arXiv 2503.12188 v2. arXiv, 12 September 2025, viewed 7 November 2026. https://arxiv.org/abs/2503.12188

24. Lee and Tiwari. Prompt Infection: LLM-to-LLM Prompt Injection within Multi-Agent Systems. arXiv 2410.07283 v1. arXiv, 9 October 2024, viewed 7 November 2026. https://arxiv.org/abs/2410.07283

25. OWASP Top 10 for Agentic Applications for 2026. OWASP Gen AI Security Project, Agentic Security Initiative, 9 December 2025, viewed 7 November 2026. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/

26. Bryan, Severi et al. Taxonomy of Failure Mode in Agentic AI Systems, whitepaper. Microsoft, April 2025, viewed 7 November 2026. https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/final/en-us/microsoft-brand/documents/Taxonomy-of-Failure-Mode-in-Agentic-AI-Systems-Whitepaper.pdf

27. New whitepaper outlines the taxonomy of failure modes in AI agents. Microsoft Security Blog, 24 April 2025, viewed 7 November 2026. https://www.microsoft.com/en-us/security/blog/2025/04/24/new-whitepaper-outlines-the-taxonomy-of-failure-modes-in-ai-agents/

28. Hammond et al. Multi-Agent Risks from Advanced AI. Cooperative AI Foundation, Technical Report #1, arXiv 2502.14143 v1. arXiv, 19 February 2025, viewed 7 November 2026. https://arxiv.org/abs/2502.14143

29. Shapira et al. Agents of Chaos. arXiv 2602.20021 v1. arXiv, 23 February 2026, viewed 7 November 2026. https://arxiv.org/abs/2602.20021

30. Hadfield, Zhang, Lien, Scholz, Fox and Ford. How we built our multi-agent research system. Anthropic Engineering, 13 June 2025, viewed 7 November 2026. https://www.anthropic.com/engineering/multi-agent-research-system

31. Building multi-agent systems: When and how to use them. Anthropic, 23 January 2026, viewed 7 November 2026. https://claude.com/blog/building-multi-agent-systems-when-and-how-to-use-them

32. Yan, W. Don't Build Multi-Agents. Cognition, 12 June 2025, viewed 7 November 2026. https://cognition.com/blog/dont-build-multi-agents

33. Yan, W. Multi-Agents: What's Actually Working. Cognition, 22 April 2026, viewed 7 November 2026. https://cognition.com/blog/multi-agents-working

34. Nechepurenko and Shuvalov. Coordination as an Architectural Layer for LLM-Based Multi-Agent Systems. arXiv 2605.03310 v1. arXiv, 5 May 2026, viewed 7 November 2026. https://arxiv.org/abs/2605.03310

About this report

Published by Rillor on 8 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Why multi-agent LLM systems fail: what the failure studies show. 8 November 2026. https://rillor.com/insights/why-multi-agent-llm-systems-fail

Notices

Superintelligence (SI) outputs can be wrong. Important decisions should include human review.

Work with Rillor

Tell us the workflow, the sources it draws on and the decisions it supports, and we scope the agents, tools and tests around them. Contact us at rillor.com/contact?type=agentic-systems or contact@rillor.com. More on agentic systems: rillor.com/si/agentic-systems