Rillor

EXPLAINER

When a multi-agent system beats a single agent

Controlled studies of single agent vs multi-agent designs find extra agents help when work splits into independent parts and hurt when each step depends on the last.



Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

- Whether a multi-agent system beats a single agent depends on the structure of the task. In 260 controlled configurations, Kim et al. (version 3, April 2026) measured relative changes in mean success from +80.8% on a decomposable financial analysis benchmark to -70.0% on a sequential planning benchmark. The mean across all six benchmarks was -0.3% [11].
- Strong single agents narrow the gain. Above about 45% single-agent accuracy, added agents lowered performance [11]. With strong prompts that include demonstrations [3], equal thinking-token budgets [12] or workflows whose agents share one base model [13], one agent matched or beat multi-agent setups. One ICLR 2026 study finds optimized multi-agent building blocks about 6% ahead [10].
- Extra agents cost more on every basis measured, from 3 to 10 times a single agent's tokens in one model developer's testing (vendor-reported) to 4 to 220 times the input tokens in one study [14, 23]. At a matched budget, successes per 1,000 tokens fell from 67.7 to between 13.6 and 42.4 [11].
- Both sides of the 2025 practitioner debate now endorse a separate checker [23, 24, 28]. The controlled evidence is mixed: Kim et al.'s error effect is not significant after controls [11], many verifiers check only the surface [8], and majority voting explains most of the gain credited to debate [5, 7].
- Data isolation, separation of duties and agents owned by other parties can require separate agents whatever the task [1, 2, 16, 19, 30]. One model developer argues that per-user permissions can be enforced inside the tools of one agent [26].
- Between June 2025 and October 2026, practitioners on both sides converged on one agent by default, with helpers that gather information [22, 23, 26, 27, 28].

Background

An agentic system uses a language model to direct its own steps. One model developer separates workflows, "systems where LLMs and tools are orchestrated through predefined code paths", from agents, "systems where LLMs dynamically direct their own processes and tool usage" [21]. Kim et al. define a single-agent system by "a solitary reasoning locus": one loop controlled by one model instance, even when the agent uses tools or reflects on its own work [11]. A multi-agent system "comprises multiple LLM-backed agents communicating through structured message passing" [11].

Multi-agent designs drew attention through 2023 studies in which model instances debated their answers [4] and through role-based frameworks such as those studied in MAST [8]. The MAST authors, analyzing more than 1,600 annotated traces from seven frameworks, write that performance gains on popular benchmarks "are often minimal" [8]. Kapoor et al. argued in 2024 that a narrow focus on accuracy had made agents "needlessly complex and costly" [15].

This report answers the single agent vs multi-agent question for engineering leads and buyers choosing between one agent with tools and a team of agents. It sets controlled results beside practitioner reports and separates task structure from the security and ownership boundaries that can require separate agents.

Method

What was examined. Comparisons of single-agent and multi-agent systems on the same tasks, debate and voting studies, a failure taxonomy, practitioner reports on their own systems, design guides and security controls.

Sources. Controlled comparisons [3, 10, 11, 12, 13, 14, 29, 30]. Debate, voting and self-correction [4, 5, 6, 7]. Failures [8], context length [9] and cost-controlled evaluation [15]. Practitioner reports from December 2024 to October 2026 [21, 22, 23, 24, 25, 26, 27, 28]. Design guidance [16, 17, 18, 20]. Security and interoperability [1, 2, 19].

How. Each figure was read in its source and is given with its date and basis. Kim et al. is cited in version 3 of 8 April 2026. Version 1, from December 2025, covered 180 configurations on four benchmarks and reported +80.9% for centralized coordination on financial reasoning [11]. Its percentages are relative: mean multi-agent success minus mean single-agent success, divided by mean single-agent success [11]. Figures that publishers measured on their own systems are labeled vendor-reported. Token multiples keep their own bases and are never pooled. Vendors and model names stay out of the text.

Findings

1. The common patterns and their token cost

The sources describe the same few patterns under different names.

Pattern	Names in the sources	Structure
Single agent with tools	Single-agent system [11, 17]	One loop plans, calls tools and acts
Orchestrator-workers	Orchestrator-workers [21], manager [20], orchestrator-subagent [24], coordinator [18], centralized [11], supervisor [30]	A lead agent splits the task, delegates parts and combines the results
Handoff	Decentralized [20], handoff [17], swarm [18, 30]	Peer agents pass control to each other
Parallel	Sectioning [21], fan-out or map-reduce [17], agent teams [24]	Independent subtasks run at once, then merge
Generator-verifier	Evaluator-optimizer [21], maker-checker [17], review and critique [18]	One agent produces, another checks against criteria
Shared conversation	Group chat or debate [17], shared state [24]	Agents build on each other's messages

One cloud guide calls a single agent with tools "often the right default for enterprise use cases" [17]. Kim et al. test four multi-agent architectures. In Independent, agents report only to an aggregator that concatenates their outputs without cross-validation or voting. In Centralized, an orchestrator talks to each agent. Decentralized is all-to-all among peers, and Hybrid adds limited peer messaging to an orchestrator [11]. Every architecture was matched to a mean of 4,800 reasoning tokens per trial [11].

All 260 configurations	Single agent	Independent	Decentralized	Centralized	Hybrid
Success rate	0.466	0.370	0.477	0.463	0.452
Turns per task	7.2	11.4	26.1	27.7	44.3
Message overhead	0%	58%	263%	285%	515%
Error amplification	1.0	17.2	7.8	4.4	5.1

All 260 configurations	Single agent	Independent	Decentralized	Centralized	Hybrid
Successes per 1,000 tokens	67.7	42.4	23.9	21.5	13.6

Source: Kim et al., Table 5, version 3, April 2026 [11].

Averaged over every task, multi-agent success rates ranged from 0.370 to 0.477 against 0.466 for the single agent, and the communicating architectures took 3.6 to 6.2 times as many turns [11]. Under fixed budgets, the authors write, per-agent reasoning capacity becomes "prohibitively thin" beyond three to four agents [11].

Other sources measure token cost on other bases, and the figures do not combine.

Source and date	Basis	Figure
Model developer, June 2025 (vendor-reported)	Against a chat interaction	About 4x for agents, 15x for multi-agent systems [22]
Model developer, January 2026 (vendor-reported)	Against a single agent on equivalent tasks	3x to 10x [23]
Gao et al., May 2025	Against a single agent, seven datasets	4x to 220x input tokens, 2x to 12x output tokens [14]

The January 2026 post traces the overhead to "duplicating context across agents, coordination messages between agents, and summarizing results for handoffs" [23]. A cloud guide suggests matching each agent's model to the complexity of its task [17]. Kim et al. found a cost to that in their setting: centralized teams that mixed models trailed teams built on the strong model alone by a mean of 12.6 percentage points [11].

In one research system, running three to five subagents in parallel, each calling three or more tools in parallel, cut research time "by up to 90% for complex queries" [22]. The same developer later wrote that multi-agent systems "often take longer overall than single-agent systems because of the sheer increase in total computation" [23]. One of its customers measured a single agent loop running about twice as fast as its multi-agent harness (vendor-reported, October 2026) [26]. Latency also builds with each handoff [16, 17].

2. Where splitting helps: breadth and parallel work

Decomposable analysis. On Finance Agent, every multi-agent architecture beat the single agent, from +57% to +80.8% relative. The centralized design reached a mean of 0.631 against 0.349 [11]. Revenue, cost and market factors "can be analyzed independently", which aligns with the coordination structure [11]. Finance

Agent and PlanCraft had similar complexity scores (0.41 and 0.42) with opposite results, so the authors conclude that sequential interdependence, more than complexity, decides whether coordination pays [11]. A decentralized design also gained 9.2% on dynamic web navigation [11].

Breadth-first research. A model developer reported in June 2025 that a multi-agent system with a frontier lead model and smaller subagents beat the lead model alone by 90.2% on its internal research evaluation (vendor-reported), with the design suited to "breadth-first queries that involve pursuing multiple independent directions simultaneously" [22]. Asked for the board members of the companies in the S&P 500 Information Technology sector, the multi-agent system decomposed the task and found them, while the single agent failed "with slow, sequential searches" [22]. Token usage by itself explained 80% of the performance variance on the BrowseComp benchmark, and the developer's own reading is that multi-agent systems "work mainly because they help spend enough tokens to solve the problem" [22].

Independent search. In August 2026 the developer's research team gave 45 agents their own virtual machines, a shared forum and the task of finding vulnerabilities in 15 open-source projects (vendor-reported) [25]. Independent parallel agents found 21 vulnerabilities over 6.5 million tokens, and the coordinating swarm found 266 over 27 million. About half of the swarm's finds lay outside the core directories, and within them the two methods "seem comparable in terms of tokens per vulnerability found" [25].

Degraded context. Across five models, performance fell by 13.9% to 85% as input grew longer, even with perfect retrieval and inputs well within the models' claimed limits [9]. Tran and Kiela show where splitting starts to pay. On four-hop questions with part of the context replaced by misleading content, the single agent led at a degradation level of 0.3, the two moved closer at 0.5, and a sequential multi-agent pipeline was clearly ahead at 0.7 [12]. A model developer lists context pollution among three situations where multiple agents outperform one [23], and a coding-agent developer notes that a read-only subagent keeps its investigative work out of the main agent's history [27].

Many tools or domains. An agent-framework maker found in February 2025 that "both more context and more tools degrade agent performance" [29]. In June 2025 it added distractor domains of 19 tools each to the retail domain of τ-bench. The single agent "falls off sharply when there are two or more distractor domains", and its token use grew with each domain while the supervisor and swarm designs stayed flat [30]. A fix to its supervisor, forwarding sub-agent answers verbatim, was among changes that yielded "a nearly 50% increase in performance" [30]. Tool-count warning levels in the guidance differ: fewer than 10 overlapping tools [20], 20 or more tools [23] and 30 to 50 tool definitions [26].

3. Where splitting hurts: sequential steps and shared decisions

Sequential tasks. On PlanCraft, a planning benchmark built on crafting tasks, every multi-agent architecture trailed the single agent: centralized by 50.3%, decentralized by 41.5%, hybrid by 39.1% and independent by 70.0% [11]. Traces show centralized systems breaking sequential tasks into "artificial subtasks", such as an

agent to check an inventory every agent could already see, and spending the token budget on coordination [11]. SWE-bench Verified showed slight losses for every multi-agent design, from -15% to -2%, and Workbench showed effects from -11% to +6% [11].

Shared decisions. A coding-agent developer argued in June 2025 that "actions carry implicit decisions, and conflicting decisions carry bad results" [27]. In its example, two subagents built parts of one product on "conflicting assumptions not prescribed upfront" [27]. It recommended a single-threaded linear agent [27]. A model developer's pattern guide says the same of agent teams: "If one teammate's work affects another's, neither is aware, and their outputs may conflict" [24]. The developer behind the research system wrote that domains with shared context or many dependencies between agents are a poor fit, and that "most coding tasks involve fewer truly parallelizable tasks than research" [22].

Splitting by role. Teams that built separate agents for planning, execution, review and iteration "suffered from lost context at each handoff and spent more tokens coordinating than executing", a model developer reported in January 2026 [23]. In one of its experiments with planner, implementer, tester and reviewer agents, "the subagents spent more tokens on coordination than on actual work" [23]. It recommends dividing work by context boundaries, so that an agent handling a feature also handles its tests [23]. A cloud adoption guide agrees that distinct roles "don't automatically justify a multi-agent architecture" [16].

The orchestrator as a bottleneck. Information one subagent finds has to travel back through the orchestrator to reach another, and after several handoffs "critical details are often lost or summarized away" [24]. Unless the subagents run in parallel, the system "incurs multi-agent token costs without the speed benefit" [24].

Practitioner positions over time. The two developers moved closer.

Date	Publisher	Position
Dec 2024	Model developer	Add complexity "only when it demonstrably improves outcomes" [21]
Jun 2025	Model developer	Multi-agent research system, 90.2% gain at about 15x chat tokens [22]
Jun 2025	Coding-agent developer	"Don't Build Multi-Agents": one single-threaded agent by default [27]
Jan 2026	Model developer	Multiple agents win in three situations, and elsewhere coordination costs typically exceed the benefits [23]
Apr 2026	Coding-agent developer	Setups work "when writes stay single-threaded" [28]

Date	Publisher	Position
Aug 2026	Model developer's research team	"True multiagent systems are still in their infancy" [25]
Oct 2026	Model developer	Fold subagents into one loop, keeping one only if folding it in scores worse [26]

The April 2026 post keeps the original objection for parallel writers and calls unstructured swarms "mostly a distraction" [28]. The October 2026 article still recommends subagents when context bloat measurably degrades one loop, when tasks need fresh context windows in parallel, or when one agent struggles to choose tools [26].

4. Strong single agents narrow the gain

The capability ceiling. Kim et al. found that tasks where a single agent already exceeds about 45% accuracy "experience negative returns from additional agents" [11]. The paper identifies this capability saturation as its robust result, confirmed under cluster-robust inference and a Holm-Bonferroni correction, and treats its other dataset-level predictors as "descriptive patterns supported by directional consistency" [11]. A rule built on its measures picked the right architecture for 87% of held-out configurations, against 20% for random choice [11].

Prompts. Wang et al. (ACL 2024) found that a single agent with strong prompts achieves almost the same performance as the leading discussion approach on a wide range of reasoning tasks, and that discussion beats a single agent "only when there is no demonstration in the prompt" [3]. With one model, averaged over three datasets, the single agent scored 75.63% with demonstrations against 74.46% for a six-agent discussion. Without demonstrations the order reversed, 67.41% against 70.75% [3]. Zhou et al. (ICLR 2026) advise to "optimize agents locally before scaling their topology" [10].

Equal budgets. Tran and Kiela (preprint, April 2026) held thinking tokens equal on two multi-hop question sets with three model families. The single agent led or was statistically indistinguishable from the leader at every budget except a 100-token one, and used fewer thinking tokens than any multi-agent variant [12]. They found artifacts in API-based budget control and in benchmarks that can inflate apparent multi-agent gains, and conclude that "many reported MAS gains are better explained by compute and context effects than by inherent architectural superiority" [12]. Xu et al. (preprint, January 2026) show across seven benchmarks that one agent can reach the performance of workflows whose agents share a base model, at lower cost through cache reuse. The result does not cover workflows that mix models [13].

Newer models and sampling. Gao et al. (preprint, May 2025) reran published comparisons with a newer model and found the accuracy gain over a single agent fell from 10% to 3%, consistently across tasks and frameworks [14]. Li et al. (TMLR 2024) found that performance scales with the number of agents through

sampling and voting alone, with larger gains on harder tasks and weaker models [6]. Kapoor et al. argue that agent evaluations "must be cost-controlled", after finding simple baselines as accurate as elaborate single-model agent designs on a coding benchmark at a fraction of the cost [15].

Practice. A model developer reported in January 2026 that teams spent months on multi-agent architectures "only to discover that improved prompting on a single agent achieved equivalent results" [23]. In October 2026 it reported that across several enterprise deployments, a single agent with skills "frequently outperformed" both a single all-purpose prompt and a subagent design on quality, often at lower cost and latency per task [26].

The counterweight. Zhou et al. found a gain of about 6% on average for optimized multi-agent building blocks over an optimized single agent, with further gains from topology and workflow-level prompts [10]. Kim et al. found large gains where tasks decompose [11].

5. Verification as a separate role

Observed error patterns. Kim et al. measured trace-level error amplification of 17.2 times for independent agents against 4.4 times for centralized coordination, where the orchestrator acts as a validation point before aggregation [11]. Architectures with an orchestrator or peer debate achieved a 22.7% average error reduction, peaking at 31.4% on Finance Agent [11]. The caveat: after controlling for other coordination metrics, the error-amplification effect does not reach statistical significance ($p = 0.658$), and efficiency and overhead explain the differences better [11].

Criteria decide the result. A model developer's pattern guide describes the generator-verifier pattern as widely deployed and warns that "the verifier is only as good as its criteria" [24]. A verifier told only to check whether output is good "will rubber-stamp the generator's output", and the pattern assumes generating and checking are separable skills [24]. The January 2026 post says the verification subagent "consistently works well across domains", since a verifier needs little context, and names checking claims or citations as a natural task for one [23]. Its main failure is a verifier that runs one or two tests, sees them pass and declares success [23]. A cloud guide asks for "clear acceptance criteria" and an iteration cap in maker-checker loops [17], and another notes that a review step adds latency and cost [18].

What failure studies show. In MAST, systems with explicit verifiers generally showed fewer total failures, but "the presence of a verifier is not a silver bullet" [8]. Many verifiers "perform only superficial checks", such as whether code compiles. Premature termination, no or incomplete verification, and incorrect verification accounted for 6.2%, 8.2% and 9.1% of annotated failures [8].

Clean-context review. A coding-agent developer reported in April 2026 that its review agent caught an average of 2 bugs per pull request even on code written by its own coding agent, "of which roughly 58% are severe" (vendor-reported), and worked better when the two agents shared no context [28]. In the June 2025 research system, a separate citation agent locates the source of each claim before results return to the user, with no measured effect reported [22].

Debate and voting. Du et al. (ICML 2024) found that three model instances debating for two rounds raised zero-shot accuracy from 67.0% to 81.8% on arithmetic and from 77.0% to 85.0% on grade-school math, against 75.0% and 81.0% for a majority vote [4]. Debate also drives agents to converge on one fact, "which is not necessarily correct" [4]. Huang et al. (ICLR 2024) replicated the experiment on the full grade-school math test set. With the same number of responses, majority voting beat debate, 85.3% against 83.2% at six responses and 88.2% against 83.0% at nine, and models "struggle to self-correct their responses without external feedback" [5]. Choi et al. (NeurIPS 2025) found across seven benchmarks that "Majority Voting alone accounts for most of the performance gains typically attributed to" multi-agent debate [7]. Wang et al. describe "wrong answer propagation", where an agent drops a correct answer to join an incorrect consensus, as a frequent error [3].

Same model, same mistakes. A model developer's research team reported in August 2026 that agents with the same context, scaffolding and model take very similar actions, so "when one agent makes a bad decision, it is likely that many agents will make that same bad decision" [25]. In one run, 18 of 30 agents created a git branch with the same name [25]. A reviewer with a clean context of its own worked better [28], and checks against external evidence avoid relying on a model to correct itself [5].

6. Boundaries that justify separate agents

Security and compliance. A cloud provider's adoption guide says to start with multiple agents when a use case crosses security and compliance boundaries, involves multiple teams or has planned growth, and otherwise to test a single agent first [16]. It recommends separate agents "when regulations or policies mandate strict data isolation", a least-privilege design that limits the blast radius of incidents [16]. Its example: "Financial services often require one agent to prepare transactions while another validates them, enforcing separation of duties through architecture" [16]. A second cloud guide lists "scenarios that require distinct security boundaries for each agent" among the cases for multiagent orchestration [17].

The controls behind them. NIST SP 800-53 describes separation of duties as addressing "the potential for abuse of authorized privileges" by dividing functions among different individuals or roles [1]. Its least privilege control allows "only authorized accesses for users (or processes acting on behalf of users) that are necessary to accomplish assigned organizational tasks" and applies to system processes too [1]. The controls are written for people, roles and processes. The cloud guides draw the analogy to agents in their own words [16, 17].

Excessive agency. OWASP defines excessive agency as the vulnerability that "enables damaging actions to be performed in response to unexpected, ambiguous or manipulated outputs from an LLM", and lists "a malicious/compromised peer agent" among its triggers in multi-agent systems [2]. It advises limiting each agent's extensions and permissions "to the minimum necessary" and enforcing authorization in downstream systems, so the model does not decide what is allowed [2]. Security surfaces expand "through extra credential management and data transit points between agents" [16], and security trimming "must be implemented in every agent in the pattern" [17].

Ownership. When another organization's agent does part of the work, "you by definition cannot have a single agent" [30]. The Agent2Agent (A2A) protocol, a Linux Foundation project, lets agents interoperate "without needing to share internal memory, tools, or proprietary logic" [19]. Separate teams are a softer case: one guide supports separate agents when "distinct teams manage separate knowledge areas" [16].

The counterpoint. A model developer argued in October 2026 that "each tool checks what the calling user may see or do and returns a plain refusal the model can read, so one loop can serve users with different access without a separate agent per permission level" [26]. It warns that one subagent per team turns the architecture into "a copy of their org chart (Conway's law at work)" and can leave "hundreds of overlapping subagents with no enforced standards and no specific owner" [26]. The positions meet at one question: does a policy require independent processing environments, which the adoption guide says single agents cannot provide, or only different access per user, which tool checks inside one loop can enforce [16, 26]?

7. A decision table

Condition	What the evidence shows	Starting design
Parts can be worked separately and share little state	+57% to +80.8% relative on Finance Agent, gains on breadth-first research (vendor-reported) [11, 22]	Orchestrator with parallel workers
Each step depends on the last	-39% to -70% relative on PlanCraft [11]	One agent
One agent already above about 45% accuracy	Negative returns from added agents [11]	One agent with improved prompts and demonstrations [3]
Compute is fixed	One agent matched or beat multi-agent designs at equal thinking-token budgets [12]	One agent
Long, noisy or misleading context	Multi-agent pipelines lead once degradation is heavy [9, 12, 23]	Read-only helpers with fresh context [27]
Many tools or overlapping domains	Single agents fall off as domains are added [29, 30]	Clarify tools first, then split by domain [20]
Output must meet explicit criteria	Verifiers help when criteria are concrete, and many checks are superficial [8, 24]	Generator plus verifier with written criteria
Several agents would write to shared work	Conflicting assumptions and outputs [24, 27]	One writer, with read-only helpers or reviewers [28]

Condition	What the evidence shows	Starting design
Isolated data, separated duties or another party's agent	Required by policy or ownership [1, 16, 30]	Separate agents with least privilege [2]
Different users need different access	Tools can enforce per-user permissions [26]	One agent with permission-checking tools

Outside policy boundaries, the guides share a first step: build one agent, test it, and move to several only "when testing reveals limitations that cannot be resolved through single-agent optimization" [16, 18, 20]. The October 2026 article adds a method: build a small end-to-end evaluation, fold in one subagent at a time, keep it separate only if the folded version scores worse, and run each task more than once [26].

Implications

For engineering leads. Keep a single-agent baseline as the comparison for every later design [16, 20]. Compare designs at equal token budgets and report cost beside accuracy [12, 15]. Split along context boundaries [23]. Give any verifier explicit criteria and evidence to check against [8, 24]. Measure the token multiple on the real workload [11, 14, 23].

For buyers. Ask what a multi-agent design scores against one agent with the same prompts and token budget [3, 12]. Ask how the work splits, whether the parts depend on each other and which agent writes the final result [11, 27, 28]. Ask what each agent may call, with what permissions, and where authorization is enforced [2, 17]. Ask what the checker checks and against which evidence [8, 24]. Treat figures such as the 90.2% gain as the publisher's own evaluation [22].

For researchers. Include majority voting at equal sample counts as a baseline for any debate method [5, 7]. Report matched-budget comparisons and repeated runs [12, 15, 26], and whether agents share a base model [13, 25].

For teams scoping a build. Rillor builds custom multi-agent systems on a client's documents, records and data. A planner splits each question into parts, each specialist agent works one part with only the tools that part needs, and an audit step checks every citation against the evidence it came from before an answer goes out. Every step is logged so a run can be replayed.

Limits of this analysis

- Kim et al. is a preprint. Two of its six benchmarks used 20-instance subsets, with bootstrap confidence intervals of about ±20 percentage points per cell, and prompts were not tuned for each model family [11]. Its 45% threshold comes from those six benchmarks.
- Kim et al.'s error-amplification figures are descriptive and not significant once other coordination metrics are controlled [11].
- The 90.2% gain, the 4x, 15x and 3x to 10x token multiples, the twofold runtime difference, the review-agent bug counts and the vulnerability counts are vendor-reported [22, 23, 25, 26, 28]. No independent replication was found.
- Most debate studies used models from 2023 to 2025 on short reasoning benchmarks without tools [3, 4, 5, 7]. Their findings do not transfer directly to orchestrator-worker systems.
- No study found here compares one agent with a planner, specialist and audit design on evidence-heavy document work such as legal research or due diligence. Among the tasks tested, Finance Agent is the decomposable analysis task that resembles such work [11].
- No source here measures maintenance effort, incident rates or security outcomes in production. The boundary guidance is design advice and control text [1, 2, 16, 17].
- Latency evidence is system-specific. Beyond Kim et al.'s turn counts, no controlled wall-clock comparison was found [11, 22, 26].
- The sources give only setting-specific agent counts: per-agent reasoning becomes "prohibitively thin" beyond three to four agents under fixed budgets [11], one research system scales from 1 agent to more than 10 subagents by query complexity [22], and one guide suggests three or fewer agents in group chat [17]. None gives a general rule.
- This explainer reports published evidence and recommends no product or vendor.

Sources

1. NIST SP 800-53 Rev. 5, Security and Privacy Controls for Information Systems and Organizations, controls AC-5 Separation of Duties and AC-6 Least Privilege. National Institute of Standards and Technology, September 2020, updated 10 December 2020, Release 5.2.0 of 27 August 2025. https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final

2. LLM06:2025 Excessive Agency, OWASP Top 10 for LLM Applications 2025. OWASP Gen AI Security Project, viewed 24 October 2026. https://genai.owasp.org/llmrisk/llm062025-excessive-agency/

3. Wang, Q., Wang, Z., Su, Y., Tong, H. and Song, Y. Rethinking the Bounds of LLM Reasoning: Are Multi-Agent Discussions the Key? Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 6106 to 6131, August 2024. https://aclanthology.org/2024.acl-long.331/

4. Du, Y., Li, S., Torralba, A., Tenenbaum, J. B. and Mordatch, I. Improving Factuality and Reasoning in Language Models through Multiagent Debate. Proceedings of the 41st International Conference on Machine Learning, PMLR 235, pp. 11733 to 11763, July 2024. https://proceedings.mlr.press/v235/du24e.html

5. Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X. and Zhou, D. Large Language Models Cannot Self-Correct Reasoning Yet. ICLR 2024, arXiv 2310.01798 version 2, 14 March 2024. https://arxiv.org/abs/2310.01798

6. Li, J., Zhang, Q., Yu, Y., Fu, Q. and Ye, D. More Agents Is All You Need. Transactions on Machine Learning Research, arXiv 2402.05120 version 2, 11 October 2024. https://arxiv.org/abs/2402.05120

7. Choi, H. K., Zhu, X. and Li, S. Debate or Vote: Which Yields Better Decisions in Multi-Agent Large Language Models? NeurIPS 2025, arXiv 2508.17536 version 2, 23 October 2025. https://arxiv.org/abs/2508.17536

8. Cemri, M., Pan, M. Z., Yang, S. and others. Why Do Multi-Agent LLM Systems Fail? NeurIPS 2025 Datasets and Benchmarks Track, arXiv 2503.13657 version 3, 26 October 2025. https://arxiv.org/abs/2503.13657

9. Du, Y., Tian, M., Ronanki, S. and others. Context Length Alone Hurts LLM Performance Despite Perfect Retrieval. Findings of EMNLP 2025, arXiv 2510.05381, 6 October 2025. https://arxiv.org/abs/2510.05381

10. Zhou, H., Wan, X., Sun, R. and others. Multi-Agent Design: Optimizing Agents with Better Prompts and Topologies. ICLR 2026, arXiv 2502.02533 version 2, 31 January 2026. https://arxiv.org/abs/2502.02533

11. Kim, Y., Gu, K., Park, C. and others. Towards a Science of Scaling Agent Systems. arXiv preprint 2512.08296 version 3, 8 April 2026. https://arxiv.org/abs/2512.08296

12. Tran, D. and Kiela, D. Single-Agent LLMs Outperform Multi-Agent Systems on Multi-Hop Reasoning Under Equal Thinking Token Budgets. arXiv preprint 2604.02460 version 2, 11 April 2026. https://arxiv.org/abs/2604.02460

13. Xu, J., Koesdwiady, A., Bei, S. and others. Rethinking the Value of Multi-Agent Workflow: A Strong Single Agent Baseline. arXiv preprint 2601.12307, 18 January 2026. https://arxiv.org/abs/2601.12307

14. Gao, M., Li, Y., Liu, B. and others. Single-agent or Multi-agent Systems? Why Not Both? arXiv preprint 2505.18286, 23 May 2025. https://arxiv.org/abs/2505.18286

15. Kapoor, S., Stroebl, B., Siegel, Z. S., Nadgir, N. and Narayanan, A. AI Agents That Matter. arXiv preprint 2407.01502, 1 July 2024. https://arxiv.org/abs/2407.01502

16. Choosing Between Building a Single-Agent System or Multi-Agent System, Cloud Adoption Framework. Microsoft Learn, updated 27 February 2026. https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ai-agents/single-agent-multiple-agents

17. AI Agent Orchestration Patterns, Azure Architecture Center. Microsoft Learn, updated 21 September 2026. https://learn.microsoft.com/en-us/azure/architecture/ai-ml/guide/ai-agent-design-patterns

18. Choose a design pattern for your agentic AI system, Cloud Architecture Center. Google Cloud, updated 28 May 2026. https://docs.cloud.google.com/architecture/choose-design-pattern-agentic-ai-system

19. A2A Protocol, project overview. The Linux Foundation, A2A Protocol project, viewed 24 October 2026. https://a2a-protocol.org/latest/

20. A practical guide to building agents. OpenAI, April 2025. https://cdn.openai.com/business-guides-and-resources/a-practical-guide-to-building-agents.pdf

21. Building effective agents. Anthropic, 19 December 2024. https://www.anthropic.com/engineering/building-effective-agents

22. How we built our multi-agent research system. Anthropic, 13 June 2025. https://www.anthropic.com/engineering/multi-agent-research-system

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

24. Multi-agent coordination patterns: Five approaches and when to use them. Anthropic, 10 April 2026. https://claude.com/blog/multi-agent-coordination-patterns

25. Patterns and problems in emerging multiagent systems. Anthropic, 13 August 2026. https://www.anthropic.com/research/multiagent-systems

26. Building agents: When to fold subagents into your main agent loop. Anthropic, 22 October 2026. https://claude.com/resources/articles/building-agents-when-to-fold-subagents-into-your-main-agent-loop

27. Yan, W. Don't Build Multi-Agents. Cognition, 12 June 2025. https://cognition.ai/blog/dont-build-multi-agents

28. Yan, W. Multi-Agents: What's Actually Working. Cognition, 22 April 2026. https://cognition.ai/blog/multi-agents-working

29. Benchmarking Single Agent Performance. LangChain, 10 February 2025. https://blog.langchain.com/react-agent-benchmarking/

30. Fu-Hinthorn, W. Benchmarking Multi-Agent Architectures. LangChain, 10 June 2025. https://blog.langchain.com/benchmarking-multi-agent-architectures/

About this report

Published by Rillor on 7 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. When a multi-agent system beats a single agent. 7 November 2026. https://rillor.com/insights/single-agent-vs-multi-agent-systems

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