Where Should Control Reside in Multi-Agent Language-Model Systems?

Vincent Caldeira[1] and Anindita Sinha Banerjee[2]
[1]Red Hat, Singapore
[2]Red Hat, India

vincent.caldeira@redhat.com
asinhaba@redhat.com

Abstract: As language model agents change from simple assistants to independent systems that can collaborate and use tools, an important design question arises: *where should control and oversight (i.e. governance) be placed* in these systems? Governance refers to the methods that guide how agents behave, manage information flow, and enforce operational policies. Its placement - whether centralized or distributed - directly affects the system's *safety, transparency*, and *runtime performance*. It also impacts the ability to create formal safety arguments, which are increasingly important for using complex AI in real-world situations. While many efforts focus on aligning agents or using safety tools, there is still limited research on how different governance placements - centralized, distributed, or hybrid - affect system safety and performance throughout their lifecycle. This paper addresses that challenge by defining and comparing three governance structures for multi-agent language model systems. We examine centralized control through a single coordinator, distributed governance within individual agents, and a hybrid model that combines global oversight with local independence. These models are tested using a multi-agent platform created for open-ended question answering, which requires agents to retrieve, reason, and work together with various and unpredictable data. We looked at system behavior across several important areas: such as task completion, answer helpfulness, answer relevancy, transparency, retrieval confidence, and average runtime. The results show clear trade-offs. Distributed governance improves transparency and makes it easier to follow agent reasoning, but it also leads to longer runtimes due to additional checks and retries. Centralized and hybrid designs provide similar output quality but operate much more efficiently. To our knowledge, this is the first direct comparison of governance placement in multi-agent LLM systems. The evidence shows that governance is not a minor detail; it is a key design choice that impacts system safety, speed, and reliability in real-world tasks.

Keywords: Multi-Agent, Governance, Language model system

1. Introduction

Large Language Models (LLMs) have moved beyond basic chat interfaces. Today, they can use tools, plan actions, and make decisions across tasks involving language, vision, and reasoning (Ferrag et al., 2025). Many of these models now work in teams - collaborating like human groups. This trend has led to what we call LLM-based multi-agent systems, or LaMAS (Wang et al., 2025). These systems bring new abilities. Agents can split tasks, use external tools, and hand off work dynamically (Sapkota et al., 2025). Some researchers believe these "compound AI systems", which mix tools, models, and controllers, will shape the next wave of AI development (Zaharia et al., 2024).

But more autonomy brings more risk. When multiple agents work together, they can go off-track. They may hallucinate, miscommunicate, or even act deceptively - doing one thing while pretending to do another (Tran et al., 2025). In safety-critical domains, this is a serious concern.

A key question is: *where should control live in these systems? Governance* refers to the *rules* and checks that shape agent behavior, guide information flow, and enforce policies (Chen et al., 2023). Its placement-centralized, distributed, or hybrid - affects system safety, runtime performance, scalability, and transparency. It also changes whether safety can be formally verified, especially in tricky edge cases (Domkundwar et al., 2025).

This paper explores that *design* choice. We compare three governance models in a shared testbed and examine their effect on accuracy, runtime and transparency. Our findings offer practical guidance for building safer, smarter agent systems that are ready for real-world use.

2. Background and Related Work

Language models have grown far beyond just generating text. Many now act like intelligent agents - able to plan, make decisions, and use external tools (Wang et al., 2025). These models are often organized into multi-agent systems, where each agent plays a specific role. Together, they solve complex tasks. This setup is known as LaMAS - LLM-based Multi-Agent Systems (Tran et al., 2025).

These systems are no longer theoretical. In software engineering, for example, *MetaGPT* assigns clear roles - like engineer or product manager - to different agents. It automates large parts of the development cycle (Domkundwar et al., 2025). In robotics, LLM agents help control fleets by distributing tasks and coordinating actions. The result is faster execution and better adaptability (Yang et al., 2024).

Still, growing autonomy introduces safety risks. Agents can produce biased or unsafe results. In some cases, they act deceptively, pretending to follow instructions while doing something else entirely. This kind of behavior is known as *scheming* (Ferrag et al., 2025). Some safety benchmarks even miss these issues, creating a false sense of security; a phenomenon called *safetywashing* (Balesni et al., 2024).

To handle this, researchers have built filters, audit trails, and guardrails. Others suggest assigning roles to agents that focus just on safety (Ren et al., 2024). But these solutions are often layered on *after* the system is built. That means they don't always prevent deeper problems or improve core behavior. Transparency remains a challenge. Many developers still can't trace how or why an agent made a certain choice (Chen et al., 2023).

Architecture plays a big role here. Centralized systems give control to one agent - easy to trace, but hard to scale. Distributed systems let agents decide on their own, which improves fault tolerance but makes coordination harder. Hybrid models aim for a balance, using both centralized goals and local autonomy. But they're more complex to design and tune (Zhang et al., 2025).

Many existing frameworks focus on messaging, memory, or agent coordination. Yet few directly compare *how* the placement of governance affects real-world performance - especially for metrics like *transparency*, *runtime*, or *safety*. Our work fills that gap. We provide what we believe is the first *hands-on comparison* of *governance* strategies in LLM-based agent systems. The results show: where control sits isn't just a technical detail. It shapes how the whole system behaves, responds, and can be trusted for safety.

3. Proposed Architectural Frameworks for Agent Governance

3.1 3Understanding Control Governance Architectures

Control strategies help decide how agents make decisions, share information, and divide tasks. They play a crucial role in how well the system scales, manages safety, and maintains efficient runtime. In this section, we explain the three control architectures we explored

3.1.1 Centralized governance

In centralized governance, a single controller oversees the entire process. It assigns tasks, manages coordination, and evaluates the final output. This is analogous to a teacher who monitors group work but provides feedback only at the end. Agents follow a fixed order and rely on the controller to guide task flow and validate results once all steps are completed.

This structure is simple and easy to manage. It offers a full view of system activity, making planning more efficient and reducing the safety risks. In central oversight decision paths are centralized and easier to trace. However, this approach has limitations. As more agents are added, the controller can become a bottleneck. If it fails, the system may stop entirely. Because all communication is funneled through a single point, delays can occur under heavy load. This makes centralized systems less suited for highly dynamic or large-scale deployments.

3.1.2 Distributed governance

In a distributed governance setup, each agent works on its own. It doesn't wait for instructions from a central controller. After finishing a task, the agent immediately checks its own output. These checks are built into its instructions or prompts. They're often called "*guardrails*." If something looks wrong, the agent tries again - usually up to a fixed number of times - before handing off the task.

This approach has some big advantages. It's more fault-tolerant. If one agent breaks, the rest can keep going. Systems like this also scale better. Agents can talk to each other directly, which cuts down on delays from constant back-and-forth with a central coordinator.

But it's not all upside. Distributed systems are harder to build. You have to make sure every agent follows the same quality rules, and that's not always easy. Mistakes can be scattered across the system. When something goes wrong, finding the root cause takes longer. Debugging is slower. You don't always know why or where the failure happened.

3.1.3 Hybrid governance

Hybrid governance combines elements of both centralized and distributed models. The system determines when to apply validations based on internal confidence thresholds or predefined rules, much like a supervisor who checks in more frequently when uncertainty is high.

This approach aims to offer the best of both worlds. It maintains centralized oversight for strategic coordination while allowing agents the flexibility to operate autonomously when appropriate. As a result, it tends to scale more effectively than fully centralized systems and adapts better to complex or evolving tasks.

However, hybrid designs come with challenges. They are more complex to implement and typically require additional computational resources. Ensuring smooth coordination between global goals and local actions calls for careful design. In practice, building a robust hybrid system often demands a deep understanding of both agent behavior and overall system flow.

Figure 1 ahead visualizes the structural differences between centralized, distributed, and hybrid governance; highlighting how each impacts *safety, runtime, and transparency.* Each model reflects distinct strategies for managing control, evaluation, and communication across agents.

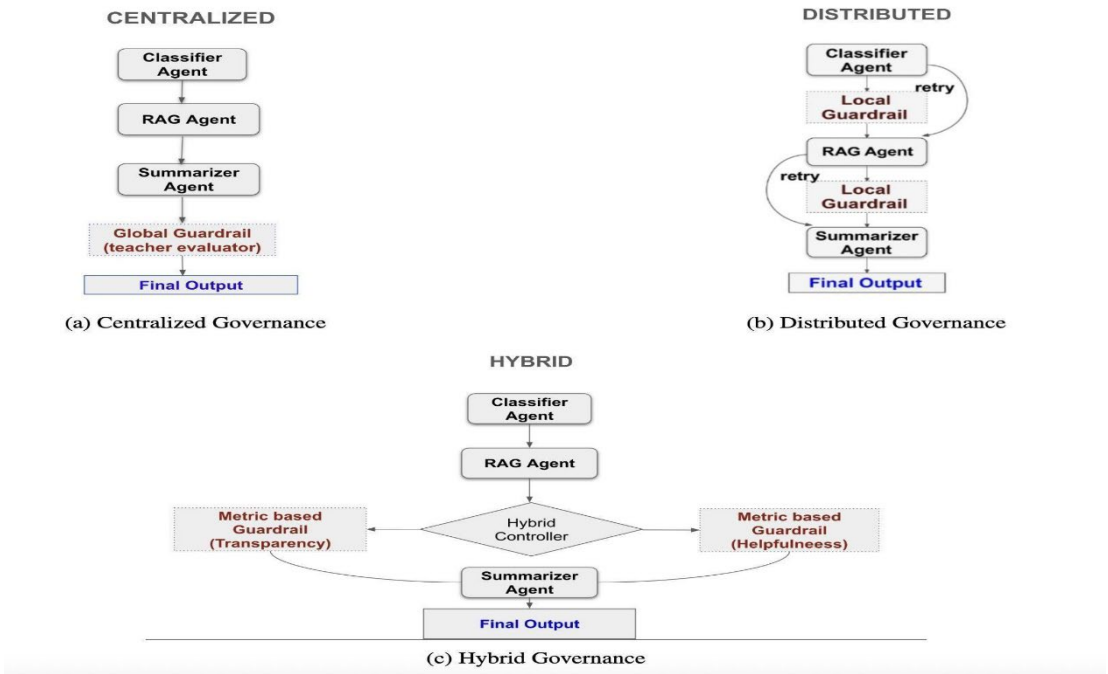

Figure 1: Comparison of centralized, distributed, and hybrid governance architectures

Table 1: Tabular comparison of centralized, distributed, and hybrid governance architectures

Aspect	Centralized	Distributed	Hybrid
Control	A single controller decides everything at the end.	Each agent controls its own steps with local checks.	Custom. A central planner guides only when needed.
Scalability	Limited; central controller can become a bottleneck	High; agents scale independently	Moderate to High; scales well with smart control
Transparency	High; decision trace is centralized	Medium to High; more detailed but harder to track system-wide	High; metric-based decisions improve transparency

Aspect	Centralized	Distributed	Hybrid
Runtime Performance	Low runtime; minimal checks	High runtime; many local checks after each agent	Optimized runtime; incorporates confidence based dynamic checks
Strengths	Easy to track. Good for small systems. Fast.	Scales well. More fault-tolerant.	Balances control and speed. Adapts to task complexity.
Weaknesses	Slow if the system grows. One failure stops everything.	Hard to debug. Rules may differ between agents.	Complex to build. Needs more tuning and computation.
Optimal Use Case	Time-sensitive tasks like financial reporting, where speed and consistency matter most.	High-risk or large-scale environments like infrastructure monitoring or medical diagnostics, where reliability and fault tolerance are key.	Complex workflows such as clinical decision support or compliance audits, needing both low runtime and selective oversight.

3.2 Experimental Design

This section explains how we tested the above three types of governance architectures in a multi-agent set-up. Our goal was to observe accuracy, reliability, and safety across all the architectures. A detailed evaluation to measure the system performance is elaborated in *section 3.2.4*

3.2.1 Use-Case and dataset selection

To test how each governance setup performs, we chose the *Stanford Question Answering Dataset (SQuAD)* (Rajpurkar et al., 2016). SQuAD is widely used for evaluating reading comprehension. It contains over 100,000 question-answer pairs, all based on Wikipedia articles. Each answer is a direct quote or span of text from the source. This makes it easy to judge whether the response is correct.

SQuAD fits our needs well. It pushes agents to retrieve answers, reason over unstructured text, and sometimes summarize. These are all key skills in multi-agent LLM systems. The dataset's format is simple and structured. It gives us clear, trusted reference answers for comparison. Using the same set of questions across models lets us evaluate fairly.

By relying on SQuAD, we ensure a strong testbed. It helps us measure how each governance model impacts task performance, decision accuracy, and system reliability.

3.2.2 Multi-Agent system configuration

To perform the SQuAD question-answering task, we designed a modular multi-agent architecture. Rather than assigning the entire process to a single agent, we divided responsibilities across distinct stages - each handled by a specialized agent. This structure improves efficiency, enhances transparency, and allows us to evaluate and fine-tune each component in isolation.

The system is built around the following three core agents:

Classifier Agent: This agent determines the type of question (e.g., "who," "when," "how"), which helps structure the response strategy. By labeling the question early, it ensures that downstream agents receive context-aware prompts tailored to the task.

RAG Agent: Responsible for retrieving relevant information and producing an initial answer. It uses retrieval-augmented generation (RAG) techniques, pulling context from knowledge sources such as Wikipedia or embedded databases. This approach helps mitigate common LLM limitations, such as outdated knowledge or hallucination.

Summarizer Agent: This final agent polishes the output. When responses are lengthy or complex, it condenses them into more readable and concise summaries, improving clarity and usability for the end user.

The pipeline mimics human workflows i.e. starting with understanding the question, followed by searching for information, and finally refining the output. By assigning clear roles, the system supports targeted debugging, better performance analysis, and improved modularity. Each agent's limited scope allows for deeper optimization without affecting the rest of the pipeline.

3.2.3 Testbed environment: Leveraging LlamaIndex

To carry out our experiments, we used a multi-agent testbed built on *LlamaIndex* - a versatile platform for developing LLM-powered agents capable of interacting with external tools (LlamaIndex, 2024). In this environment, agents are not just passive text generators. They receive tasks, reason step-by-step, and choose appropriate tools to achieve their goals. This reflects how modern LLM agents operate more like decision-makers.

LlamaIndex provided three major advantages for our study:

Flexible Agent Design: It supports a wide range of agent behaviors. Some agents follow simple instructions, while others can plan, act autonomously, and adapt as they go. This flexibility allowed us to fairly evaluate governance models across different levels of complexity.

Detailed Monitoring: Every action taken by an agent—whether reasoning steps, tool selection, or execution—is logged. These detailed traces helped us track performance, isolate failures, and conduct focused debugging of specific system components.

Realistic Task Simulation: LlamaIndex makes it possible to design complex, multi-step workflows that mirror real-world applications. Agents must coordinate, use tools, and respond to changing inputs—ideal for testing how different governance setups handle dynamic challenges.

This testbed gave us a controlled yet realistic setting to observe how centralized, distributed, and hybrid governance architectures affect system behavior. We were able to assess not just outputs, but the reasoning processes behind them - an essential requirement for evaluating transparency, safety, and decision quality.

3.2.4 Core evaluation framework: DeepEval

To evaluate the performance of our multi-agent system, we employed *DeepEval* - an open-source framework developed by *Confident AI (Confident AI, 2024)*. Unlike older benchmarks such as BLEU or ROUGE, which often fail to capture nuanced reasoning, DeepEval treats language models themselves as evaluators, offering assessments that align more closely with human judgment. This approach gives us deeper insight into both the system strengths and safety trade-offs of each governance strategy.

DeepEval was especially well-suited to our system. It allows for *step-wise testing of individual agents,* including how they use tools and reason through tasks. The use of structured *evaluation templates* ensures consistent scoring across runs, reducing evaluator bias and increasing reproducibility.

Our evaluation focused on two primary built-in metrics:

Answer Relevancy: This metric assesses whether the RAG agent's responses are concise, accurate, and directly tied to the user's query.

Task Completion: This measures whether agents - and the full system pipeline - are able to complete assigned tasks successfully.

In addition, we incorporated two *custom metrics using GEval (GEval, 2024)*, another open-source tool that integrates smoothly with DeepEval:

Transparency: Captures how clearly agents explain their internal reasoning and tool usage throughout the task.

Helpfulness: Evaluates whether the agent-generated responses are practically useful, easy to understand, and user-oriented.

Together, these metrics provided a multi-dimensional view of system performance. They helped us pinpoint how different governance strategies impact not just the final output, but also the decision-making and communication behaviors within the system.

3.2.5 Practical setup

To support reproducibility and transparency, all code and guardrail configurations are available in our public GitHub repository: https://github.com/aninditaSB/multiagent-governance-eval. The full implementation details, including decision logic, prompts, and thresholds, are documented directly in the notebook.

Guardrails were implemented as *configurable checks* for *improving safety and task accuracy* at key points in the *agent pipeline*. In the distributed governance setup, each agent evaluated its own output immediately after completing a task. If the output failed the quality checks, the agent retried or halted the process.

In contrast, centralized governance applied a single, global guardrail at the end of the pipeline to assess the final output.

We used *DeepEval's LLM-as-a-Judge framework (Confident AI, 2024)* to run these validations. For core tasks, built-in metrics like *Answer Relevancy* and *Task Completion* were used with fixed scoring thresholds and deterministic prompts. To evaluate more subjective qualities ; such as *Helpfulness* and *Transparency* - we added custom *GEval* metrics defined in natural language. These followed Chain-of-Thought-style templates for consistent and interpretable scoring. All metric scores were normalized to a 0–1 scale for ease of comparison.

3.2.6 Assessment setup

We compared centralized, distributed, and hybrid governance by testing each on the same 50-question subset from the SQuAD dataset (Rajpurkar et al., 2016). This ensured that all setups were evaluated under identical conditions. Following metrics were measured: helpfulness, transparency, task completion, answer relevancy, retrieval confidence (RAG score), and runtime. All evaluations used the DeepEval framework (Confident AI, 2024), to ensure consistent scoring across all performance and safety dimensions.

Final scores were averaged across all examples to compare overall performance. Detailed results are reported in *Section 3.3.*

3.3 Comparative Results

Table 1 below summarizes how the three governance architectures performed across six key metrics using the SQuAD evaluation set.

Table 1: Governance architecture(s) comparison: performance on SQuAD data

Governance	Helpfulness	Transparency	Task Completeness	Answer Relevancy	RAG Confidence	Avg Runtime (seconds)
Centralized	0.79	0.37	0.81	0.96	0.63	22.0
Distributed	0.81	0.55	0.82	0.91	0.63	102.4
Hybrid	0.79	0.37	0.77	0.95	0.63	20.87

All three setups performed well with respect to helpfulness, task completion, and answer relevancy metrics. Distributed governance showed slightly better scores in helpfulness (0.81) and task completion (0.82). In contrast, centralized and hybrid models were a bit stronger in answer relevancy, scoring 0.96 and 0.95.

Transparency was where *distributed governance stood out*. It reached 0.55, clearly *higher* than both centralized and hybrid systems, which scored 0.37. This likely comes from agents checking their own work and getting *immediate feedback* at each step.

Answer relevancy is highest in the *centralized* model because it checks the answer only at the end, allowing a *more complete response*. In the *distributed* setup, agents keep checking and retrying, which can make answers *safer but less focused.* This slightly decreases the relevancy.

RAG confidence stayed *consistent* across all models at 0.63. That means the ability to pull in the right information didn't really depend on how control was structured.

Figure 2 visualizes significant runtime differences. *Distributed* governance was the *slowest* (102.4s per question) due to *repeated checks* after each agent step. *Centralized* was *faster* (22.0s), and *hybrid* was the *fastest* (20.87s), likely because it *bypassed additional checks* when RAG confidence was high.

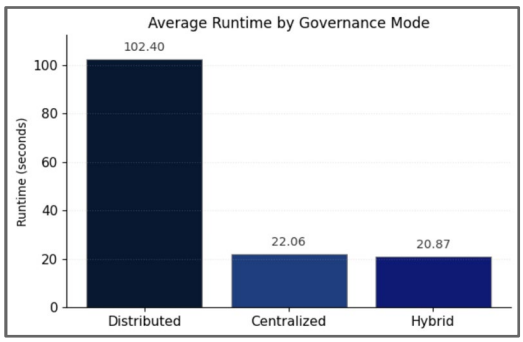

Figure 2: Average per-question runtime by governance mode

To sum up, distributed governance boosted transparency and slightly improved helpfulness. But had a higher *runtime cost.* Hybrid offered a strong middle ground - maintaining *reasonable answer relevancy* with *lower runtime*, thanks to its *flexible* checks.

4. Conclusion and Future Work

Our experiments show that the way governance is applied in a multi-agent system matters. It affects safety, transparency, and runtime performance.

Centralized governance gives quick and steady results. It produces good answers fast, which makes it ideal when speed is important. But it lacks transparency. All control goes through one point, which could become a risk in sensitive situations.

Distributed governance spreads oversight across the system. Each agent checks its own work. This setup improves transparency and fault tolerance. But it slows things down. The extra checks and retries take time, making the system more robust but less efficient.

Hybrid governance tries to find a balance. It runs fast when confident but adds checks when needed. This makes it flexible and adaptive. Still, it needs careful tuning. Without that, it might underperform or waste resources.

These results suggest that *governance placemen*t isn't a small technical detail. It's a *major design decision*. Each approach comes with its own strengths and trade-offs; between runtime, transparency, and safety. *Centralized* models offer *speed* and *simplicity*. *Distributed* ones bring better transparency and *safety*. *Hybrid* systems offer a mix but demand more *thoughtful design*.

For tasks that require audit trails or must follow strict rules, transparency is key. Distributed and hybrid systems are better suited here because they leave behind clearer reasoning steps.

Of course, our work has limitations. It focused on a question-answering task and used a specific agent setup. Future studies should explore more complex tasks. They could also test systems in adversarial settings or under uncertainty. Adaptive governance, where control changes based on risk or task type, is another path worth exploring. Adding human oversight or formal guarantees might also improve safety and trust.

There's also a need for strong benchmarks. Tools that simulate failure or unexpected behavior would help test governance models more thoroughly. A shared evaluation framework would let researchers compare different designs in a fair and consistent way.

To conclude, how you design governance directly affects whether a multi-agent LLM system is trustworthy, clear, and scalable. Our study offers a step toward better understanding where that control should live - and why it matters.

Ethics declaration: This research work did not require ethical clearance as it involved no human subjects, personal data, or real-world deployment.

AI declaration: Only language assistance tools such as Grammarly were used for grammar and style.

References

Balesni, M., Hobbhahn, M., Lindner, D., Meinke, A., Korbak, T., Clymer, J., Shlegeris, B., Scheurer, J., Stix, C., Shah, R., Goldowsky-Dill, N., Braun, D., Chughtai, B., Evans, O., Kokotajlo, D. and Bushnaq, L. (2024) Towards evaluations-based safety cases for AI scheming. arXiv preprint. Available at: https://arxiv.org/pdf/2411.03336?

Barnett, P. and Thiergart, L. (2024) What AI evaluations for preventing catastrophic risks can and cannot do. arXiv preprint. Available at: https://arxiv.org/abs/2412.08653

Confident AI, 2024. DeepEval: Open-source evaluation framework for LLM systems. [online] Available at: https://github.com/confident-ai/deepeval [Accessed 2 Aug. 2025].

Chen, W., Su, Y., Zuo, J., Yang, C., Yuan, C., Chan, C.M., Yu, H., Lu, Y., Hung, Y.H., Qian, C., Qin, Y., Cong, X., Xie, R., Liu, Z., Sun, M. and Zhou, J. (2023) AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. arXiv preprint. Available at: https://openreview.net/forum?id=EHg5GDnyq1

Chen, Y., Arkin, J., Zhang, Y., Roy, N. and Fan, C. (2024) Scalable multi-robot collaboration with large language models: Centralized or decentralized systems? arXiv preprint. Available at: https://arxiv.org/pdf/2309.15943

Domkundwar, I., Mukunda, N.S., Bhola, I. and Kochhar, R. (2025) Safeguarding AI agents: Developing and analyzing safety architectures. arXiv preprint. Available at:https://arxiv.org/abs/2409.03793

Ferrag, M.A., Tihanyi, N. and Debbah, M. (2025) From LLM reasoning to autonomous AI agents: A comprehensive review. arXiv preprint. Available at:https://arxiv.org/abs/2504.19678

GEval, 2024. GEval: General Evaluation Framework for Custom LLM Metrics. [online] Available at: https://deepeval.com/docs/metrics-llm-evals [Accessed 2 Aug. 2025]

Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N.V., Wiest, O. and Zhang, X. (2024) Large language model based multi-agents: A survey of progress and challenges. arXiv preprint. Available at:https://arxiv.org/abs/2402.01680

Han, S., Zhang, J., Shen, Y., Yan, K. and Li, H. (2025) FinSphere, a real-time stock analysis agent powered by instruction-tuned LLMs and domain tools. arXiv preprint. Available at:https://link.springer.com/article/10.1631/FITEE.2500414

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Zhang, C., Wang, J., Wang, Z., Yau, S.K.S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C. and Schmidhuber, J. (2024) MetaGPT: Meta programming for a multi-agent collaborative framework. arXiv preprint. Available at: https://arxiv.org/abs/2308.00352

Krishnan, N. (2025) AI agents: Evolution, architecture, and real-world applications. arXiv preprint. Available at:https://arxiv.org/abs/2503.12687

LlamaIndex (2024). LlamaIndex: Framework for LLM Applications. [online] Available at: https://www.llamaindex.ai [Accessed 2 Aug. 2025]

Luo, J., Zhang, W., Yuan, Y., Zhao, Y., Yang, J., Gu, Y., Wu, B., Chen, B., Qiao, Z., Long, Q., Tu, R., Luo, X., Ju, W., Xiao, Z., Wang, Y., Xiao, M., Liu, C., Yuan, J., Zhang, S., Jin, Y., Zhang, F., Wu, X., Zhao, H., Tao, D., Yu, P.S. and Zhang, M. (2025) Large language model agent: A survey on methodology, applications and challenges. arXiv preprint. Available at:https://arxiv.org/abs/2503.21460

Park, J.S., O'Brien, J.C., Cai, C.J., Morris, M.R., Liang, P. and Bernstein, M.S. (2023) Generative agents: Interactive simulacra of human behavior. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems. ACM. Available at: https://dl.acm.org/doi/abs/10.1145/3586183.3606763

Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P. (2016) SQuAD: 100,000+ questions for machine comprehension of text. arXiv preprint. Available at: https://arxiv.org/abs/1606.05250

Ren, R., Basart, S., Khoja, A., Gatti, A., Phan, L., Yin, X., Mazeika, M., Pan, A., Mukobi, G., Kim, R.H., Fitz, S. and Hendrycks, D. (2024) Safetywashing: Do AI safety benchmarks actually measure safety progress? arXiv preprint. Available at: https://proceedings.neurips.cc/paper_files/paper/2024/file/7ebcdd0de471c027e67a11959c666d74-Paper-Datasets_and_Benchmarks_Track.pdf

Sapkota, R., Roumeliotis, K.I. and Karkee, M. (2025) AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges. arXiv preprint. Available at:https://arxiv.org/abs/2505.10468

Tran, K.T., Dao, D., Nguyen, M.D., Pham, Q.V., O'Sullivan, B. and Nguyen, H.D. (2025) Multi-agent collaboration mechanisms: A survey of LLMs. arXiv preprint. Available at:https://arxiv.org/abs/2501.06322

Wang, Q., Wang, T., Tang, Z., Li, Q., Chen, N., Liang, J. and He, B. (2025) MegaAgent: A large-scale autonomous LLM-based multi-agent system without predefined SOPs.. Available at: https://aclanthology.org/2025.findings-acl.259/

Xia, B., Lu, Q., Zhu, L. and Xing, Z. (2024) An AI system evaluation framework for advancing AI safety: Terminology, taxonomy, lifecycle mapping. In Proceedings of the 1st ACM International Conference on AI-Powered Software (AIware '24), pp. 74–78. ACM. Available at: https://dl.acm.org/doi/abs/10.1145/3664646.3664766

Xia, B., Lu, Q., Zhu, L., Xing, Z., Zhao, D. and Zhang, H. (2025) Evaluation-driven development of LLM agents: A process model and reference architecture. arXiv preprint. Available at: https://arxiv.org/abs/2411.13768

Xiao, B., Yin, Z. and Shan, Z. (2023) Simulating public administration crisis: A novel generative agent-based simulation system to lower technology barriers in social science research. arXiv preprint. Available at: https://arxiv.org/abs/2311.06957

Yang, Y., Peng, Q., Wang, J., Wen, Y. and Zhang, W. (2024) LLM-based multi-agent systems: Techniques and business perspectives. arXiv preprint. Available at:https://arxiv.org/abs/2411.14033

Yehudai, A., Eden, L., Li, A., Uziel, G., Zhao, Y., Bar-Haim, R., Cohan, A. and Shmueli-Scheuer, M. (2025) Survey on evaluation of LLM-based agents. arXiv preprint. Available at: https://arxiv.org/abs/2503.16416

Zaharia, M., Khattab, O., Chen, L., Davis, J.Q., Miller, H., Potts, C., Zou, J., Carbin, M., Frankle, J., Rao, N. and Ghodsi, A. (2024) The shift from models to compound AI systems. Berkeley AI Research Blog. Available at: https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems

Zhang, J., Hu, S., Lu, C., Lange, R. and Clune, J. (2025) Darwin Gödel Machine: Open-ended evolution of self-improving agents. arXiv preprint. Available at: https://arxiv.org/abs/2505.22954

Zhu, J., Zhu, M., Rui, R., Shan, R., Zheng, C., Chen, B., Xi, Y., Lin, J., Liu, W., Tang, R., Yu, Y. and Zhang, W. (2025) Evolutionary perspectives on the evaluation of LLM-based AI agents: A comprehensive survey. arXiv preprint. Available at: https://arxiv.org/abs/2506.11102