International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Orchestrating Multi-Agent AI Systems: Guardrails, Trust Boundaries, and Coordination Patterns for Enterprise Deployments

Arjun Danda Sureshbabu

Abstract: Enterprise AI has shifted from single-model, single-task systems toward architectures in which specialized agents collaborate, delegate, and coordinate to accomplish complex multi-step business tasks , handling finance reporting, operations workflows, and compliance checks through automated agent chains. This shift introduces a class of reliability and governance failures that individual agent quality improvements cannot address: trust boundary violations between agents, failure propagation through agent chains, resource contention on shared data, and the progressive erosion of human oversight as automation depth increases. This paper presents a principled orchestration framework for enterprise multi-agent AI systems, comprising three architectural elements , explicit trust boundary models, a two-layer guardrail architecture operating at both agent and orchestration levels, and a coordination pattern taxonomy covering sequential delegation, parallel specialization, and hierarchical orchestration. A progressive automation governance framework is developed for expanding autonomous agent scope incrementally as operational confidence grows. The analysis argues that reliable enterprise multi-agent deployment depends on governance infrastructure , trust contracts, guardrail ownership, audit trails , as much as on individual agent capability, and that organizations deploying multi-agent systems without this infrastructure will encounter failure modes that model improvements alone cannot resolve.

Keywords: *AI Agent Orchestration, Enterprise AI, Guardrails, Multi-Agent Systems, Trust Boundaries*

1. Introduction

Finance AI workflows at Snowflake that process revenue recognition through multi-step agent chains encounter a failure mode absent from single-agent architectures: a malformed output from an upstream agent is consumed by a downstream agent without validation, the downstream agent builds on it, and by the time the error surfaces to a human reviewer, it has propagated through multiple processing steps and become difficult to trace to its origin [8]. Both agents operated within their individual capability limits. The failure occurred at the coordination layer , in the absence of a trust boundary requiring the consuming agent to validate what it received before building on it.

Prompt engineering, output validation, and RLHF improve individual agent reliability by making agents less likely to produce incorrect outputs [9]. These approaches do not prevent a downstream agent from misusing a correct-but-out-of-context output, or prevent two agents from producing conflicting modifications to shared state, or preserve human oversight when a multi-agent workflow automates a sequence of decisions that individually appear routine but collectively constitute a high-stakes business action [2]. These are coordination-layer problems requiring coordination-layer solutions.

Three architectural elements address the problem at the layer where it originates. Explicit trust boundaries define what agents are permitted to do with each other's outputs and what verification receiving agents must apply before acting on what they receive. A two-layer guardrail architecture governs both individual agent outputs and agent interaction combinations that produce policy-relevant aggregate outcomes. Coordination patterns provide structured approaches to agent delegation, parallel specialization, and hierarchical orchestration that build human oversight into the workflow architecture rather than treating it as a post-deployment addition [4].

This paper presents these three elements as a unified orchestration framework for enterprise multi-agent AI systems. The analysis argues that reliable multi-agent deployment is as much a governance problem as a technical one , the trust contracts, audit trails, and guardrail ownership structures that sustain reliable single-agent deployments must be extended to govern the interactions between agents in multi-agent architectures.

Independent Researcher, USA

2. Related Works

2.1 Multi-Agent AI Systems: Architectures and Challenges

Interleaving reasoning traces with tool invocations , the architectural insight of the ReAct framework , gave LLM-based agents the ability to plan and execute multi-step tasks by alternating between thinking and acting rather than committing to a plan before execution begins [1]. Scaling this architecture to multi-agent systems followed naturally: tasks too complex for a single agent could be decomposed into subtasks handled by specialized agents coordinated by an orchestrator, producing capability gains on software engineering, research synthesis, and complex workflow automation tasks that single-agent systems cannot complete reliably [3], [4]. The foundational theory of intelligent agent behavior and multi-agent organizations provides the conceptual grounding for understanding how authority flows through hierarchical agent systems [12], [13].

Enterprise deployment introduces requirements that capability benchmarks do not surface. Auditability , the ability to reconstruct which agent took which action, on what input, at what confidence level , is a compliance requirement in regulated industries before it is a feature preference [11]. Predictable failure modes , knowing how the system behaves when an agent produces an incorrect output rather than discovering it after propagation , are a reliability requirement for business-critical workflows [7]. Human oversight preservation , ensuring that automated agent chains do not collectively constitute a high-stakes decision without human review , is a governance requirement that most multi-agent frameworks treat as optional configuration [2]. Large-scale generative agent simulations demonstrate that agent coordination behavior at scale surfaces emergent governance challenges absent in small-scale deployments [14].

Trust and delegation in multi-agent AI has been examined primarily through the lens of agent cooperation capability: can agents share information effectively, coordinate tasks without conflict, and achieve complex goals collaboratively [10]. Practical frameworks such as LangChain and AutoGPT have demonstrated the engineering feasibility of autonomous multi-agent pipelines [16], [17], while theoretical work on cooperative AI identifies open governance problems at the agent interaction layer [18]. The enterprise production question is structurally different: can an organization trust that a multi-agent workflow will not take policy-violating actions, produce misleading outputs, or fail in ways that are auditable and recoverable? Existing trust frameworks address the first question without systematically addressing the second [11].

2.2 Guardrails and Safety in LLM-Based Systems

Constitutional AI methods that train models to critique and revise their own outputs against principled constraints established a foundational approach to agent behavioral alignment [2]. Runtime enforcement frameworks extend this to execution time , AgentSpec provides a domain-specific language for specifying behavioral constraints enforced during agent operation, preventing unsafe executions in over 90% of code agent cases tested and eliminating all hazardous actions in embodied agent task benchmarks [5]. Guard agent architectures deploy a dedicated oversight agent to monitor target agent actions against defined safety policies, with GuardAgent achieving guardrail accuracy above 98% on healthcare access control benchmarks [7].

Each of these approaches constrains what individual agents can do without constraining what combinations of agents can do together. An orchestration-level policy violation arises when Agent A's output is compliant, Agent B's output is compliant, but Agent B acting on Agent A's output produces an aggregate outcome that violates enterprise policy , a combination that neither agent-level guardrail detects because each monitors only individual agent outputs [4]. The two-layer guardrail architecture in this paper addresses this gap by adding an orchestration layer governing agent interactions alongside the agent-level constraints that existing approaches provide.

Human oversight in AI systems has been approached through human-in-the-loop mechanisms routing uncertain or high-stakes decisions to reviewers. Multi-agent architectures introduce a distinct oversight preservation challenge: as automation depth increases, the number of agent actions between human-visible checkpoints grows, and the aggregate business impact of those automated actions may exceed what any single action would trigger for review [9]. Building escalation paths into the coordination pattern architecture , rather than relying on human reviewers to identify when intervention is warranted , is the structural approach the coordination pattern taxonomy in this paper provides. Agent frameworks

that incorporate verbal reinforcement and self-reflection mechanisms offer additional pathways for oversight-preserving agent behavior [21].

3. Methods and Material

3.1 Trust Boundary Architecture

Three tiers classify inter-agent relationships by the verification requirements a receiving agent must apply to a sending agent's output before acting on it. Trusted peer relationships apply between agents deployed under the same governance context , sharing validation standards, output schema contracts, and organizational accountability , and require only lightweight schema validation before output consumption. Verified delegation relationships apply when a downstream agent receives outputs that may have been produced under different governance conditions, requiring output schema validation and semantic consistency checks before the receiving agent acts. Untrusted input relationships apply when an agent receives outputs originating outside the governed multi-agent system , external APIs, user-provided content, cross-organizational agents , requiring full validation equivalent to external data ingestion before use [11].

Table 1. Trust Boundary Tiers: Relationship Type, Verification Requirements, Failure Isolation Properties

Trust Tier	Relationship Type	Verification Requirements	Failure Isolation Property	Example Enterprise Context
Tier 1 , Trusted Peer	Agents deployed together under the same governance context, sharing validation standards and schema contracts	Lightweight output schema validation; no full semantic verification required	Failures contained within the peer agent pair; shared governance context enables rapid joint rollback	Two agents in the same Snowflake finance workflow sharing the same data access policies and output schema contracts
Tier 2 , Verified Delegation	Downstream agent receiving outputs from an upstream agent that may have been produced under different governance conditions	Output schema validation AND semantic consistency check before acting on received output	Verification boundary prevents upstream failure from propagating; downstream agent can reject non-conforming outputs and escalate	A coordinator agent delegating to a specialized reporting agent with different data access scope and prompting strategy
Tier 3 , Untrusted Input	Agent receiving outputs originating outside the governed multi-agent system: external APIs, user-provided content, cross-organizational agents	Full validation equivalent to external data ingestion: schema validation, semantic verification, provenance check, cryptographic attestation for high-stakes chains	Strict validation boundary; untrusted inputs treated as external data until fully validated; non-conforming inputs rejected before any processing	A finance agent receiving data from an external market data API or a user-provided report to be ingested into an automated workflow

Three implementation mechanisms enforce trust boundaries at runtime. Output schemas constrain what a sending agent may produce, enforced at the output layer before transmission. Input validators constrain what a receiving agent will accept, enforced at the input layer before processing. Cryptographic attestation provides provenance verification for high-stakes delegation chains, ensuring the output being consumed is genuinely the output of the claimed source agent and has not been modified in transit [5]. Together, these mechanisms convert abstract trust tier classifications into enforceable runtime constraints that do not depend on agent-level cooperation to function.

3.2 Layered Guardrail Architecture

Layer 1, agent-level guardrails, constrains individual agents , limiting output scope, action space, and confidence thresholds. This is the

guardrail category addressed by existing single-agent safety approaches [7]. Layer 1 prevents individual agents from producing out-of-scope outputs, taking actions outside their permitted space, or proceeding when output confidence falls below a governance threshold. It is necessary but insufficient for enterprise multi-agent reliability, because individual agent compliance does not guarantee compliant agent combinations.

Layer 2, orchestration-level guardrails, governs agent interactions invisible to individual agent-level monitoring. Rate limiting prevents automated workflows from accumulating actions faster than human oversight can track , an orchestration constraint that no individual agent can enforce, since each sees only its own actions. Policy combination constraints check that the aggregate effect of multiple agents' outputs does not violate enterprise policy even when each output individually complies [4]. Human approval gates enforce review requirements for action combinations crossing a defined risk threshold, regardless of whether any individual action would independently trigger review.

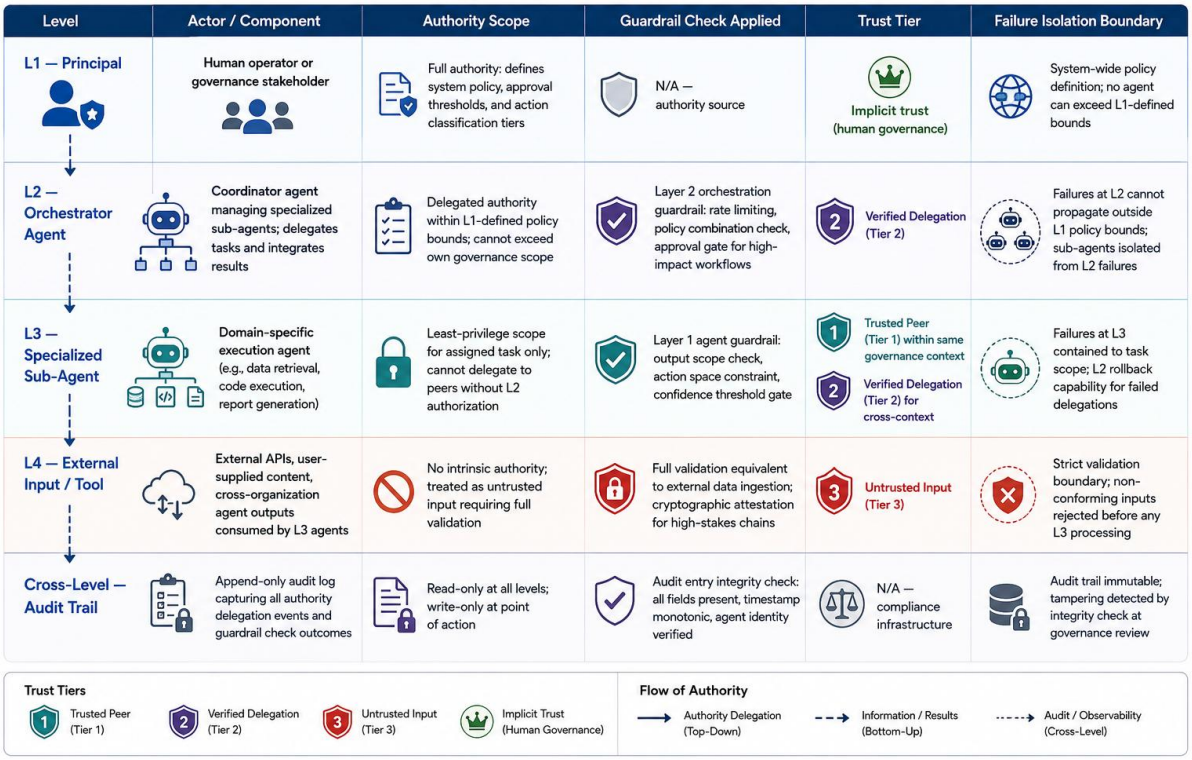


Figure 1. Trust and Authority Flow in a Hierarchical Multi-Agent Enterprise System

Guardrail calibration is a governance process, not a technical configuration. Parameters , thresholds, scope constraints, policy combination rules , are derived from enterprise compliance requirements and operational risk assessments. As agent capabilities evolve and policies change, the governance owners who defined the parameters must update them. A static guardrail system will be progressively bypassed by capability improvements its parameters do not anticipate [2].

4. Results and Discussion

4.1 Coordination Pattern Taxonomy

Sequential delegation , Agent A completing a task and passing its output to Agent B, which builds on it , is the most common and most failure-prone multi-agent coordination structure in enterprise deployments [10]. Reliability requirements span three levels. At the handoff layer, Agent A's output must be validated against Agent B's input schema before the handoff completes, preventing Agent B from receiving structurally malformed inputs that produce unpredictable behavior. At the semantic consistency layer, Agent B must verify not just structural validity but coherence with its task context , that the output it received is what the workflow context led it to expect. At the orchestrator recovery layer, the orchestrating system must maintain rollback capability to Agent A's state when Agent B's processing reveals an Agent A error, enabling

workflow recovery without manual state reconstruction [5].

Table 2. Coordination Pattern Taxonomy: Pattern Type, Agent Relationship, Failure Mode, Orchestration Requirement

Pattern	Agent Relationship	Primary Failure Mode	Orchestration Requirement	Enterprise Use Case
Sequential Delegation	Agent A completes task; passes output to Agent B which builds on it	Error propagation: Agent A's incorrect output consumed by Agent B without validation, compounding before surfacing	Handoff schema validation; semantic consistency check; orchestrator rollback capability to Agent A state	Finance pipeline: data extraction agent → calculation agent → report generation agent
Parallel Specialization	Multiple specialized agents execute concurrently; aggregator agent synthesizes outputs	Conflicting output: specialized agents produce contradictory assessments of overlapping data domains	Confidence-weighted aggregation; scope enforcement preventing overlapping data domain access; conflict escalation path	Due diligence pipeline: financial analysis agent + risk analysis agent + compliance check agent → synthesis agent
Hierarchical Orchestration	Coordinator agent manages collection of specialized sub-agents; delegates tasks and integrates results	Delegation authority gap: coordinator delegation decisions are opaque and not auditable; sub-agent conflicts unresolvable by coordinator	Auditable delegation log; sub-agent output validation before coordinator acts; defined escalation path to human review for unresolvable conflicts	Operations center: coordinator agent delegating monitoring, alerting, and remediation to specialized sub-agents across service domains

Parallel specialization , multiple agents executing concurrently on different aspects of a task, with an aggregator synthesizing outputs , introduces failure modes orthogonal to sequential delegation. Conflicting output is the most consequential: two specialized agents producing contradictory assessments of overlapping data domains, with the aggregator receiving inputs it cannot synthesize without either choosing arbitrarily or escalating. Confidence-weighted aggregation manages this by mapping each specialized agent's output to a calibrated confidence score, preferring higher-confidence outputs in conflicts , but this requires comparable confidence scoring across agents, which mandates governance over how each specialized agent expresses uncertainty [15]. Orchestration-level scope enforcement prevents conflicting outputs by preventing agents from operating on overlapping data domains without explicit coordination.

Hierarchical orchestration , a coordinator agent managing specialized sub-agents , aligns most closely with enterprise organizational management structures and most directly raises governance questions about delegation authority. Coordinator delegation decisions must be auditable: each delegation logged with the triggering reasoning, the confidence level at which the coordinator committed to the delegation, and the expected output format the sub-agent is contracted to produce. Sub-agent outputs must be validated against coordinator expectations before the coordinator acts on them. Escalation paths must be defined for cases where sub-agent conflicts or confidence gaps exceed defined thresholds , routing to human review for cases the coordinator cannot resolve algorithmically [11]. Multi-agent collaboration frameworks have demonstrated that structured role assignment and emergent behavior monitoring are both essential governance levers in hierarchically coordinated systems [22].

4.2 Human Oversight and Governance Framework

The progressive automation framework provides a governance model for expanding multi-agent automation scope incrementally as operational confidence is earned rather than assumed. Stage 1 , supervised automation , requires human approval before every agent action executes, providing complete human control at the cost of eliminating automation throughput benefits. Stage 1 is the appropriate starting point for any new multi-agent workflow, enabling operations teams to observe agent behavior and build confidence before extending autonomous scope. Stage 2 , monitored automation , allows agent actions to execute autonomously with real-time human visibility and override capability, preserving the ability to intervene without requiring approval for every action. Stage 3 , audited automation , allows fully autonomous execution with post-hoc audit and periodic governance review, appropriate for workflows where extended Stage 2 operation has built demonstrated confidence in the system's judgment [9].

Table 3. Progressive Automation Framework: Stage, Agent Autonomy Level, Human Oversight Mechanism, Advancement Criteria

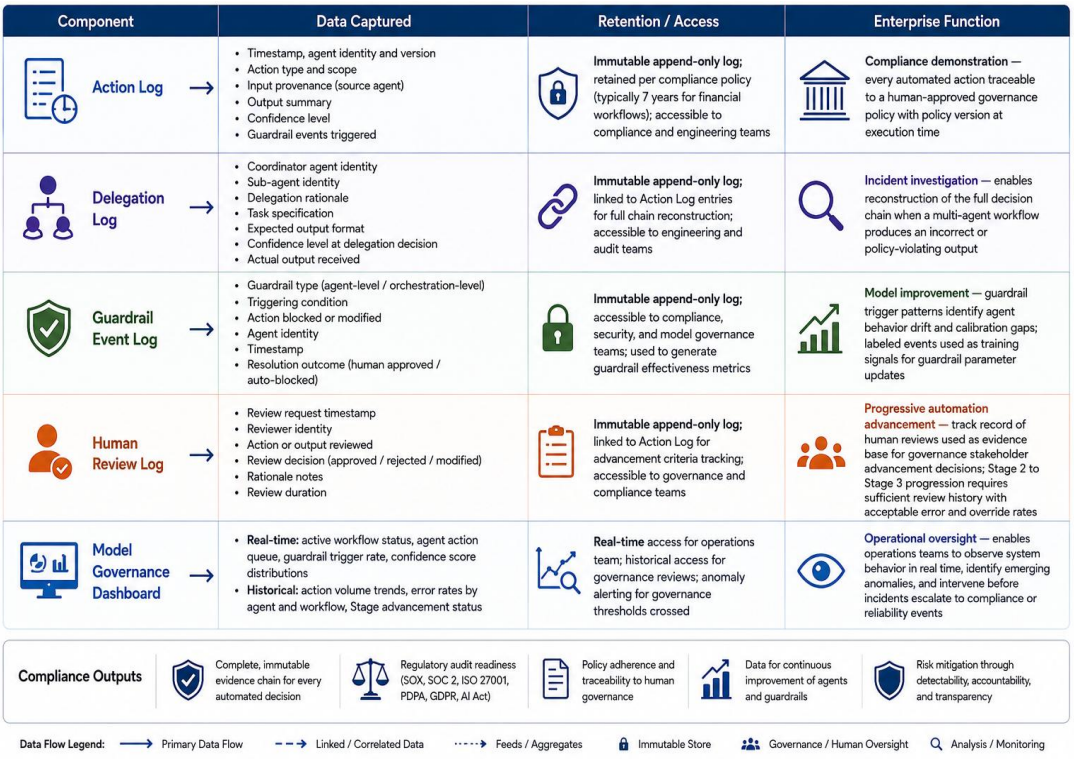
Stage	Agent Autonomy Level	Human Oversight Mechanism	Throughput Characteristic	Advancement Criteria
Stage 1 , Supervised Automation	No autonomous action; every agent action requires human approval before execution	Human approval gate before every action; full real-time visibility; override capability at all times	Low , limited by human approval throughput; appropriate for initial deployment and building operational confidence	Operations team has reviewed sufficient action volume to characterize system behavior; responsible governance stakeholders approve advancement based on track record review
Stage 2 , Monitored Automation	Actions execute autonomously with real-time human visibility and override capability	Real-time monitoring dashboard; override capability for flagged actions; notifications for actions crossing defined risk thresholds	Medium , automation throughput with human oversight safety net; appropriate for workflows with demonstrated Stage 1 reliability	Sustained operation at Stage 2 with error rate and override rate below governance thresholds; governance stakeholders approve advancement after track record review period (typically 30-90 days)
Stage 3 , Audited Automation	Full autonomous execution with post-hoc audit and periodic governance review	Post-hoc audit log review; periodic governance review (weekly/monthly depending on risk level); exception alerting for anomalous action patterns	High , full automation throughput; appropriate for workflows with demonstrated Stage 2 reliability and low organizational risk tolerance for the action types	Governance stakeholders confirm regulatory compliance of automated workflow; compliance team has reviewed audit trail quality; incident response procedures are documented and tested

Advancement criteria , the conditions governing progression between stages , are governance decisions reflecting organizational risk tolerance, regulatory requirements, and demonstrated operational track record, not technical accuracy milestones [11]. Treating advancement as a

technical threshold rather than a governance decision creates accountability gaps that surface as attribution failures when incidents occur.

Every agent action, delegation decision, and inter-agent output exchange must be logged with five fields: timestamp, agent identity and version, action type and scope, input provenance, and outcome. This audit trail serves three enterprise functions simultaneously. Compliance demonstration traces every automated action to a human-approved governance policy with the policy version active at execution time. Incident investigation enables reconstruction of the full decision chain when a multi-agent workflow produces an incorrect or policy-violating output. Model improvement labels agent action outcomes as training signals for guardrail calibration and agent fine-tuning that build reliability from production evidence. Comprehensive surveys of LLM-based agent capabilities highlight the importance of structured observation and audit infrastructure as preconditions for responsible agent deployment at enterprise scale [23].

Figure 2. Multi-Agent Audit Trail Architecture: Components, Data Flows, and Compliance Outputs



Conclusions

Multi-agent AI architectures introduce coordination-layer reliability and governance problems that individual agent quality improvements cannot resolve. Trust boundary violations, failure propagation through agent chains, and the erosion of human oversight as automation depth increases are structural properties of the multi-agent coordination architecture , not deficiencies of individual agents. They require structural solutions at the orchestration layer: trust boundary models that enforce verification requirements between agents, a two-layer guardrail architecture that governs both individual agent behavior and agent interaction combinations, and coordination patterns that build human oversight into workflow structure.

The governance argument is inseparable from the technical one. A multi-agent system deployed with trust boundary contracts but without governance processes keeping those contracts current as agents evolve will find its boundaries progressively misaligned. A guardrail system deployed without active ownership of its calibration parameters will be bypassed by capability improvements it was never updated to constrain. A progressive automation framework adopted as a technical milestone rather than a governance process will advance workflows to full automation before the operational confidence that justifies advancement exists [11]. Technical architecture and governance infrastructure must develop together.

Enterprise AI architects designing multi-agent systems for finance, operations, and compliance workflows should implement trust boundary contracts, the two-layer guardrail architecture, and structured coordination patterns as first-class design elements from the outset , not as post-deployment additions when their absence has already produced incidents. The progressive automation framework provides the governance model for expanding autonomous scope as operational confidence is earned, building organizational trust in multi-agent systems through demonstrated reliability rather than through assumption.

References

[1] Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K, et al. ReAct: Synergizing reasoning and acting in language models. arXiv, 2023. Available from: https://arxiv.org/abs/2210.03629

[2] Bai Y, Jones A, Ndousse K, Askell A, Chen A, DasSarma N, et al. Constitutional AI: Harmlessness from AI feedback. arXiv preprint arXiv:2212.08073; 2022. Available from: https://arxiv.org/abs/2212.08073

[3] Wang L, Ma C, Feng X, Zhang Z, Yang H, Zhang J, et al. A survey on large language model based autonomous agents. arXiv, 2025. Available: https://arxiv.org/pdf/2308.11432

[4] He J, Treude C, Lo D. LLM-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. ACM Trans Softw Eng Methodol. 2025;34. Availble from: https://dl.acm.org/doi/10.1145/3712003

[5] Wang H, Poskitt CM, Sun J. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. In: Proceedings of ICSE 2026; 2025. Available from: https://arxiv.org/pdf/2503.18666

[6] Asai A, Wu Z, Wang Y, Sil A, Hajishirzi H. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. arXiv; 2023. Available from: https://arxiv.org/pdf/2310.11511

[7] Xiang Z, Zheng L, Zheng Z, Li B. GuardAgent: Safeguard LLM agents by a guard agent via knowledge-enabled reasoning. arXiv, 2025. Available from: https://arxiv.org/pdf/2406.09187

[8] Ji Z, Lee N, Frieske R, Yu T, Su D, Xu Y, et al. Survey of hallucination in natural language generation. arXiv, 2024. Available from: https://arxiv.org/pdf/2202.03629

[9] Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C, Mishkin P, et al. Training language models to follow instructions with human feedback. arXiv, 2022. Available from: https://arxiv.org/pdf/2203.02155

[10] Guo T, Chen X, Wang Y, Chang R, Peng S, Chawla NV, et al. Large language model based multi-agents: A survey of progress and challenges. IJCAI '24: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, 2024. Available from: https://dl.acm.org/doi/10.24963/ijcai.2024/890

[11] Raza S, et al. TRiSM for Agentic AI: A review of trust, risk, and security management in LLM-based agentic multi-agent systems. arXiv, 2025. Available from: https://arxiv.org/pdf/2506.04133

[12] Wooldridge M, Jennings NR. Intelligent agents: theory and practice. Cambridge University Press, 2009. Available from: https://www.cambridge.org/core/journals/knowledge-engineering-review/article/abs/intelligent-agents-theory-and-practice/CF2A6AAEEA1DBD486EF019F6217F1597

[13] Ferber J, Gutknecht O, Michel F. From agents to organizations: an organizational view of multi-agent systems. Agent-Oriented Software Engineering IV, 2003. Available from: https://link.springer.com/chapter/10.1007/978-3-540-24620-6_15

[14] Park JS, O'Brien JC, Cai CJ, Morris MR, Liang P, Bernstein MS. Generative agents: interactive simulacra of human behavior. arXiv, 2023. Available from: https://arxiv.org/pdf/2304.03442

[15] Shen Y, Song K, Tan X, Li D, Lu W, Zhuang Y. HuggingGPT: solving AI tasks with ChatGPT and its friends in HuggingFace. Arxiv, 2023. Available from: https://arxiv.org/pdf/2303.17580

[16] Chase H. LangChain: building applications with large language models. GitHub repository. 2022. https://github.com/langchain-ai/langchain

[17] Significant Gravitas. AutoGPT: an autonomous GPT-4 experiment. GitHub repository. 2023. https://github.com/Significant-Gravitas/AutoGPT

[18] Dafoe A, Hughes E, Bachrach Y, Collins T, McKee KR, Leibo JZ, et al. Open problems in cooperative AI. arXiv; 2020. Available from: https://arxiv.org/pdf/2012.08630

[19] Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. arXiv, 2021; Available from: https://arxiv.org/pdf/2005.11401

[20] Zhao WX, Zhou K, Li J, Tang T, Wang X, Hou Y, et al. A survey of large language models. arXiv preprint arXiv:2303.18223; 2026. Available from: https://arxiv.org/pdf/2303.18223
[21] Shinn N, Cassano F, Gopinath A, Narasimhan K, Yao S. Reflexion: language agents with verbal reinforcement learning. arXiv, 2023. Available from: https://arxiv.org/pdf/2303.11366
[22] Chen W, Su Y, Zuo J, Yang C, Yuan C, Chan CM, et al. AgentVerse: facilitating multi-agent collaboration and exploring emergent behaviors. arXiv, 2023. Available from: https://arxiv.org/pdf/2308.10848
[23] Xi Z, Chen W, Guo X, He W, Ding Y, Hong B, et al. The rise and potential of large language model based agents: a survey. arXiv; 2023. Available from: https://arxiv.org/pdf/2309.07864