Global Research Journal in Engineering, Science, Technology and Management (GRJESTM)
Volume 02, Issue 03 (May-June) 2026, pp. 361-364
GRJESTM :: An odaswa™ Journal
ISSN : 3107-3611 (Online)

TECHNICAL SERIES | TECHNOLOGY INSIGHT – TN07

Agentic AI and Multi-Agent Systems: From Language Models to Autonomous Software Workflows

Rohan Malhotra[1]* and Ananya Kapoor[2]

Technology Snapshot.

Technology Domain	Artificial Intelligence and Software Engineering
Core Technologies	Large Language Models (LLMs), tool calling, planning, memory and workflow orchestration
System Architecture	Single-agent and multi-agent architectures
Primary Applications	Software development, research assistance, IT operations, business process automation
Key Challenges	Reliability, security, coordination, cost and evaluation
Technology Maturity	Rapidly evolving; production suitability depends on the application and level of human oversight

Keywords: Agentic AI, multi-agent systems, large language models, autonomous workflows, tool use, orchestration, AI safety.

* 1. Department of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab, India.
2. Department of Computer Science and Engineering, Amity University, Noida, Uttar Pradesh, India.

Copyright: © 2026 Rohan Malhotra and Ananya Kapoor. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. Published By **odaswa**. Published in Chennai, Tamil Nadu, India

1. Introduction

Large Language Models (LLMs) have expanded the capabilities of AI systems in language understanding, code generation and information synthesis. However, conventional LLM applications generally respond to individual prompts and depend on external software or human users to execute subsequent actions. Agentic AI extends this interaction model by enabling an AI system to plan tasks, invoke tools, observe results and revise its actions toward a specified objective.

An AI agent combines a reasoning model with an execution environment, tools, state management and control logic. Instead of generating only a final response, it can perform a sequence of operations, such as retrieving documents, executing code, validating outputs and preparing a structured report. Multi-agent systems extend this concept by assigning complementary responsibilities to multiple agents and coordinating their activities.

These capabilities are increasingly relevant to software engineering and enterprise automation. Nevertheless, agentic systems should not be interpreted as inherently reliable or fully autonomous: their performance depends on task decomposition, available tools, context quality, verification mechanisms and operational constraints.

2. Operating Principle and System Architecture

An agentic workflow typically follows an iterative cycle of goal interpretation, planning, action, observation and evaluation. The system first converts a user objective into manageable subtasks. It then selects an appropriate tool or operation, executes the action and examines the resulting observation. Depending on the result, the agent may continue, revise its plan, request clarification or terminate the workflow.

A representative architecture contains the following components:

- **Reasoning engine:** Interprets instructions, generates plans and selects candidate actions.
- **Task planner:** Decomposes a complex objective into ordered or dependent subtasks.
- **Tool interface:** Connects the agent to APIs, databases, search services, code interpreters and enterprise applications.
- **Memory and state management:** Maintains relevant conversation context, task progress and selected persistent information.
- **Execution controller:** Enforces permitted actions, iteration limits, timeouts and workflow transitions.
- **Verification layer:** Checks intermediate results and final outputs against explicit requirements.
- **Human oversight:** Supports approval, clarification, exception handling and review of high-impact actions.

The distinction between a language model and an agent is therefore architectural rather than merely linguistic. The model supplies reasoning and generation capabilities, while the surrounding software enables controlled interaction with external systems.

3. Single-Agent and Multi-Agent Workflows

A single agent can manage a complete task through repeated tool calls. This approach is relatively straightforward to implement and monitor, making it suitable for bounded tasks with limited dependencies. More complex workflows may benefit from multiple specialized agents, each assigned a distinct role. Comparison of workflow architectures is shown in Table 1.

Table 1. Comparison of workflow architectures

Feature	Single-agent system	Multi-agent system
Task execution	One agent manages the workflow	Multiple agents share responsibilities
Coordination	Comparatively simple	Requires message passing and orchestration
Specialization	Roles are handled by one agent	Roles can be separated by expertise or function
Resource consumption	Often lower	Can increase through parallel calls and repeated context
Failure handling	Centralized debugging	Failures may propagate between agents
Suitable applications	Bounded, sequential tasks	Complex tasks with separable subtasks

For example, an automated software-maintenance workflow could use a planning agent to interpret an issue, a coding agent to propose a change, a testing agent to execute tests, and a review agent to inspect the patch. A coordinator manages task dependencies and combines the results. This separation can improve modularity, but it does not automatically guarantee better accuracy.

Multi-agent designs may use centralized orchestration, in which a coordinator assigns tasks, or decentralized coordination, in which agents communicate and negotiate responsibilities. Parallel execution can reduce elapsed time when subtasks are independent, whereas sequential dependencies and communication overhead may eliminate this advantage.

4. Implementation Considerations

Developing an agentic workflow requires more than connecting an LLM to a set of tools. Reliable implementation depends on clear task boundaries, constrained permissions, observable execution and measurable completion criteria.

4.1. Task decomposition and control

Objectives should be divided into verifiable subtasks. Maximum iteration counts, execution timeouts and explicit stopping conditions help prevent repeated or unproductive action loops.

4.2. Tool integration

Tools should expose well-defined interfaces, validated inputs and predictable outputs. Read-only access should be preferred where possible. Actions involving file deletion, financial transactions, external publication or changes to production systems should require appropriate authorization.

4.3. Context and memory management

Large task histories can increase processing costs and introduce irrelevant information. Systems should retain task-relevant state, distinguish verified facts from agent-generated assumptions, and avoid storing sensitive information unnecessarily.

4.4. Validation and observability

Each important operation should produce logs containing the action, input, output, status and relevant error details. Testing should measure task completion, factual accuracy, tool-call success,

recovery from errors, latency and cost. Human review remains important when failures could cause significant consequences.

4.5. Security

Agentic workflows can be exposed to prompt injection, malicious retrieved content, excessive tool permissions and unintended disclosure of confidential data. External content should be treated as untrusted input. Permission boundaries, secret management, sandboxing and human approval for sensitive actions reduce—but do not eliminate—these risks.

5. Technical Challenges and Emerging Directions

The principal challenge is ensuring that an agent achieves the intended outcome rather than merely producing plausible intermediate reasoning. A mistaken early action can influence later decisions, while errors can accumulate across long workflows. Multi-agent systems introduce additional concerns, including conflicting outputs, duplicate work, communication overhead and ambiguous responsibility for failures.

Current engineering approaches emphasize constrained execution, structured tool interfaces, explicit state machines, independent verification and evaluation using realistic task scenarios. Rather than relying exclusively on an agent's self-assessment, systems can use deterministic tests, schema validation, policy checks and independent reviewers to verify critical outputs.

Emerging directions include agents that operate across software-development environments, agents that collaborate through standardized tool interfaces, and workflow systems that combine LLM-based reasoning with conventional business rules. A likely development path is toward bounded autonomy: agents receive greater freedom for low-risk, reversible tasks while high-impact operations remain subject to approval and auditable controls.

6. Conclusion

Agentic AI represents a transition from prompt-based language-model interaction to goal-oriented software workflows involving planning, tool execution, observation and verification. Multi-agent architectures extend this approach by distributing work among specialized components, particularly when subtasks can be separated meaningfully. Their practical value depends not simply on model capability, but on robust orchestration, security controls, reliable evaluation and effective human oversight. In production systems, autonomy should therefore be designed as a controlled engineering capability rather than assumed as an inherent property of an AI model.

References

[1] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu and J. Wen, "A Survey of Large Language Models," arXiv preprint arXiv:2303.18223, 2023.

[2] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," International Conference on Learning Representations (ICLR), 2023.

[3] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," arXiv preprint arXiv:2308.08155, 2023.

[4] Y. Shinn, B. Labash, P. Gopinath, K. Narasimhan and A. Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," Advances in Neural Information Processing Systems (NeurIPS), Volume 36, 2023.