Agentic AI: A Review of Autonomous AI Agents

RINKU[1], ARYAN[2], JUNED[3], PARAS[4], JYOTI BHARDWAJ[5]
[1, 2, 3, 4, 5]*Department of Computer Science and Engineering (AI & ML), Panipat Institute of Engineering and Technology (PIET), Samalkha, Panipat, Haryana, India*

Abstract- ***Autonomous artificial intelligence (AI) agents, systems capable of perceiving an environment, reasoning about goals, planning multi-step actions, invoking external tools, and adapting based on feedback, have emerged as a dominant paradigm for extending the capabilities of large language models (LLMs) beyond single-turn text generation. Termed agentic AI, this paradigm encompasses single-agent architectures with planning and memory modules as well as multi-agent systems in which specialized agents collaborate, negotiate, or compete to accomplish complex objectives. This review surveys the foundations of agentic AI, including planning strategies such as ReAct, Tree-of-Thoughts, and reflection-based self-correction; memory architectures spanning short-term context windows, vector-based long-term memory, and episodic memory stores; and tool-usage frameworks that allow agents to call external application programming interfaces (APIs), execute code, and query databases. We examine prominent multi-agent frameworks including LangGraph, CrewAI, and AutoGen, comparing their orchestration models, communication protocols, and suitability for different task structures. We then survey applications of agentic AI in healthcare, finance, and education, highlighting both demonstrated benefits and domain-specific risks. Finally, we discuss open challenges including reliability and hallucination propagation across agent chains, coordination overhead in multi-agent systems, security vulnerabilities such as prompt injection, and the evaluation gap for long-horizon autonomous behavior, before outlining directions for future research.***

Index Terms—Agentic AI, Autonomous Agents, Multi-Agent Systems, LLM Planning, Tool Use, Memory Architectures, Langgraph, Crewai, Autogen, AI Safety.

I. INTRODUCTION

Large language models have progressed from simple text completion engines to systems capable of sophisticated multi-step reasoning, tool invocation, and goal-directed behavior. This evolution has given rise to the field of agentic AI, in which an LLM is embedded within a broader control loop that allows it to observe an environment, decide on an action, execute that action through an external interface, and incorporate the resulting observation into subsequent decisions. Unlike a conventional chatbot that produces a single response to a single prompt, an autonomous agent operates over an extended horizon, potentially issuing dozens or hundreds of intermediate actions before arriving at a final result.

The appeal of agentic AI lies in its ability to automate complex, multi-step workflows that previously required substantial human orchestration: booking travel across multiple constraints, conducting multi-source research and synthesis, debugging and refactoring large software repositories, or coordinating clinical decision support across multiple specialties. At the same time, the extended autonomy and tool access granted to agents introduces new categories of risk, including error propagation across long action chains, security vulnerabilities from untrusted tool outputs, and challenges in evaluating success for open-ended tasks.

This paper aims to provide a comprehensive review of the agentic AI landscape as of the current generation of frameworks and deployed systems. We organize the review around four pillars that recur across nearly all agentic architectures: planning, memory, tool usage, and multi-agent coordination. We then survey real-world applications and conclude with a discussion of open research challenges.

The remainder of this paper is structured as follows. Section II reviews the historical and technical background of AI agents, including classical planning and reinforcement learning approaches that preceded LLM-based agents. Section III examines planning strategies specific to LLM agents. Section IV discusses memory architectures. Section V covers tool usage and function calling. Section VI surveys multi-agent orchestration frameworks. Section VII discusses domain applications in healthcare, finance, and education. Section VIII presents a comparative

discussion of the surveyed frameworks. Section IX outlines open challenges and future directions, and Section X concludes the paper.

II. BACKGROUND AND RELATED WORK

The concept of an autonomous software agent predates modern LLMs by decades. Classical AI planning systems, formulated using representations such as STRIPS and the Planning Domain Definition Language, allowed a system to search over a space of possible action sequences to reach a specified goal state given a symbolic model of the world. Reinforcement learning (RL) later provided a framework for agents to learn action policies directly from interaction with an environment through trial and reward, achieving notable success in game-playing domains.

The introduction of instruction-tuned LLMs shifted the locus of reasoning from hand-crafted symbolic planners or learned RL policies toward natural-language reasoning performed by a pretrained model. Early demonstrations showed that prompting an LLM to produce a chain of intermediate reasoning steps before a final answer, commonly termed chain-of-thought prompting, substantially improved performance on multi-step arithmetic and logical reasoning tasks. This observation directly motivated subsequent work on structuring LLM outputs as explicit action-observation loops, giving rise to the modern notion of an LLM-based agent.

Related surveys have examined narrower aspects of this space, such as tool-augmented language models or retrieval-augmented generation; this review differentiates itself by focusing specifically on the orchestration layer that binds planning, memory, and tool use into autonomous, extended-horizon behavior, and by providing a comparative treatment of the dominant open-source multi-agent frameworks in current use.

III. PLANNING STRATEGIES FOR LLM AGENTS

A. ReAct: Reasoning and Acting

The ReAct framework interleaves free-text reasoning traces with discrete actions, prompting the model to alternate between a 'thought' step that reflects on the current state and an 'action' step that invokes a tool or produces an output. The resulting observation from the environment is appended to the context, and the loop repeats until a termination condition is met. This tight interleaving of reasoning and action has become the de facto baseline pattern underlying most modern agent frameworks.

B. Tree-of-Thoughts and Deliberate Search

Whereas ReAct produces a single linear reasoning trajectory, Tree-of-Thoughts and related deliberate search methods allow the agent to explore multiple candidate reasoning branches in parallel, evaluate the promise of each partial solution using a self-assessment or scoring function, and backtrack from unpromising branches. This more closely resembles classical search-based planning and has demonstrated improved performance on tasks with combinatorial solution spaces, such as puzzle solving and complex multi-constraint scheduling, at the cost of substantially higher inference-time compute.

C. Reflection and Self-Correction

Reflection-based methods introduce an explicit self-critique step in which the agent reviews its own prior output or action outcome, identifies errors or shortcomings, and revises its subsequent plan accordingly. This has been shown to reduce certain classes of errors, particularly those stemming from misapplied assumptions, though reflection is less effective at correcting errors rooted in genuine knowledge gaps rather than reasoning slips.

D. Hierarchical and Task-Decomposition Planning

For long-horizon tasks, many agentic systems adopt a hierarchical structure in which a high-level planner decomposes an overarching goal into an ordered set of subtasks, each of which may be delegated to a specialized sub-agent or executed by a lower-level planning loop. This decomposition mirrors classical hierarchical task network planning and helps constrain the context and tool access required at each step, improving both reliability and interpretability of the overall agent trajectory.

IV. MEMORY ARCHITECTURES FOR AUTONOMOUS AGENTS

A. Short-Term Context Memory

The most basic form of agent memory is the model's own context window, which retains the full sequence of prior thoughts, actions, and observations within a single episode. While simple to implement, this approach is constrained by the finite context length of the underlying LLM and incurs quadratic computational cost as the transcript grows, motivating summarization or truncation strategies for long-running agents.

B. Vector-Based Long-Term Memory

To persist information beyond a single episode or context window, many agent frameworks incorporate an external vector database, such as ChromaDB, FAISS, or Pinecone, into which relevant facts, prior conversation summaries, or task outcomes are embedded and stored. At inference time, a retrieval step fetches the most semantically relevant memory entries given the current context, which are then injected back into the prompt. This retrieval-augmented memory pattern allows an agent to accumulate knowledge across sessions without requiring model retraining.

C. Episodic and Working Memory

Some agent architectures explicitly distinguish between working memory, holding information relevant to the immediate subtask, and episodic memory, recording a structured log of completed tasks, decisions, and their outcomes for later reflection or auditing. This separation supports more sophisticated behaviors such as learning from past mistakes across multiple task attempts and generating human-readable audit trails of agent decision-making.

V. TOOL USAGE AND FUNCTION CALLING

A defining characteristic of agentic AI is the ability to invoke external tools, including web search, code execution environments, database queries, and third-party APIs, as part of its reasoning loop. Modern LLM providers expose structured function-calling interfaces in which the model is presented with a schema describing available tools, their parameters, and expected return types, and the model emits a structured call rather than free-form text when it determines a tool invocation is required.

Effective tool usage depends on several factors: the clarity and specificity of tool descriptions provided to the model, the model's ability to correctly format structured arguments, and the surrounding orchestration logic that validates tool outputs before returning them to the agent's reasoning loop. Malformed or hallucinated tool calls remain a common failure mode, particularly for tools with complex nested parameter schemas or ambiguous natural-language descriptions.

Beyond single tool calls, many agentic systems support parallel tool invocation, where multiple independent tool calls are issued simultaneously to reduce latency, and tool chaining, where the output of one tool call is programmatically routed as input to another, often orchestrated by a control framework such as LangGraph rather than left entirely to model-driven decision-making.

VI. MULTI-AGENT ORCHESTRATION FRAMEWORKS

A. LangGraph

LangGraph models an agentic workflow as a directed graph in which nodes represent discrete processing steps, potentially including individual LLM calls, tool invocations, or conditional branching logic, and edges represent control flow between steps. This graph-based abstraction gives developers fine-grained control over execution order, supports cyclic graphs for iterative refinement loops, and integrates with persistent checkpointing to allow long-running or resumable agent workflows.

B. CrewAI

CrewAI adopts a role-based abstraction in which a set of specialized agents, each assigned a distinct role, goal, and backstory, collaborate on a shared objective under the coordination of a process manager that determines task sequencing. This framework is particularly well suited to workflows that map naturally onto a team-of-specialists metaphor, such as a research agent, a writing agent, and an editing agent collaborating on a document.

C. AutoGen

AutoGen frames multi-agent interaction as a conversation between multiple LLM-backed agents, each of which can be configured with distinct system prompts, tool access, and termination conditions. Agents communicate by exchanging messages within a shared conversational thread, and the framework supports both fully autonomous multi-agent conversations and human-in-the-loop configurations where a human participant can intervene at designated checkpoints.

D. Comparative Remarks

LangGraph offers the most explicit control over execution flow and is well suited to production systems requiring deterministic behavior and observability. CrewAI offers a higher-level, role-oriented abstraction that accelerates prototyping of collaborative workflows at some cost to fine-grained control. AutoGen's conversational abstraction is particularly effective for tasks that benefit from iterative back-and-forth refinement between agents, such as code generation followed by code review.

VII. APPLICATIONS OF AGENTIC AI

A. Healthcare

In healthcare, agentic systems have been proposed to assist with clinical documentation, triage support, and multi-specialty coordination, where a primary agent gathers patient information and delegates specific diagnostic sub-questions to specialized reasoning modules. Reported benefits include reduced documentation burden on clinicians and faster synthesis of multi-source patient data, though deployment remains constrained by the need for rigorous validation, regulatory approval, and clear accountability structures when an autonomous system contributes to clinical decisions.

B. Finance

Financial applications of agentic AI include automated research synthesis across earnings reports and market data, multi-step portfolio analysis, and customer service agents capable of executing account actions through banking APIs. The high stakes and regulatory scrutiny associated with financial decision-making have led most production deployments to constrain agent autonomy through mandatory human approval steps for any action with material financial consequence.

C. Education

In education, agentic tutoring systems have been explored to decompose a student's learning goal into a sequence of practice tasks, monitor performance across sessions using episodic memory, and adaptively adjust difficulty or explanation style. Multi-agent configurations, in which one agent generates practice problems and another critiques student responses, have shown promise for personalized feedback generation, though questions remain regarding the accuracy and pedagogical soundness of AI-generated feedback compared to expert human tutors.

VIII. COMPARATIVE DISCUSSION

Across the surveyed planning strategies, a consistent trend emerges: methods offering greater reasoning flexibility, such as Tree-of-Thoughts and hierarchical decomposition, tend to improve task success rates on complex problems at the cost of substantially increased inference latency and token consumption relative to simpler linear approaches such as ReAct. This trade-off suggests that production deployments should select planning strategies based on task complexity and latency tolerance rather than defaulting uniformly to the most sophisticated available method.

Similarly, memory architecture choice should be matched to task horizon: short-lived, single-session tasks are adequately served by context-window memory alone, whereas applications requiring persistence across sessions, such as personalized tutoring or long-term customer relationship management, benefit substantially from vector-based long-term memory despite the added infrastructure complexity of maintaining an external memory store.

Among multi-agent frameworks, no single framework dominates across all criteria; the choice between LangGraph, CrewAI, and AutoGen is best understood as a trade-off between control granularity, development speed, and suitability to conversational versus workflow-graph task structures.

IX. OPEN CHALLENGES AND FUTURE DIRECTIONS

Despite rapid progress, agentic AI faces several unresolved challenges. Error propagation remains a central concern: because agents operate over extended action sequences, a single incorrect intermediate decision, such as a misinterpreted tool output, can compound across subsequent steps and lead to a substantially degraded final outcome, a failure mode with no direct analogue in single-turn LLM usage.

Security is a further pressing concern, particularly prompt injection attacks in which adversarial content embedded within a retrieved document, web page, or tool output is crafted to hijack the agent's subsequent behavior. Because agents by design incorporate untrusted external content into their reasoning context, they present a substantially larger attack surface than a closed single-turn chatbot.

Coordination overhead in multi-agent systems, including redundant computation, miscommunication between agents operating on inconsistent shared state, and difficulty debugging emergent multi-agent behavior, remains a practical engineering challenge without a fully standardized solution. Evaluation methodology also lags behind system capability: existing benchmarks largely measure success on short, well-defined tasks, while real-world agentic deployments increasingly involve open-ended, long-horizon objectives for which ground-truth success criteria are difficult to specify.

Promising directions for future work include formal verification techniques adapted for agent action sequences, standardized long-horizon evaluation benchmarks with graduated task complexity, hybrid neuro-symbolic architectures that combine LLM-based reasoning with verifiable symbolic planners for safety-critical subtasks, and improved sandboxing and permissioning frameworks that constrain the blast radius of any individual agent action.

X. EVALUATION BENCHMARKS FOR AGENTIC SYSTEMS

Rigorous evaluation of agentic AI systems requires benchmarks that go beyond single-turn question answering to capture multi-step task completion, tool-use correctness, and robustness to distractor or adversarial content encountered mid-task. Several benchmark suites have emerged to address this gap. WebArena and its successors evaluate agents on realistic web-browsing tasks, requiring multi-step navigation, form completion, and information synthesis across simulated e-commerce, forum, and content-management websites, with success measured against a verified end-state rather than surface-level output similarity.

SWE-bench evaluates coding agents on their ability to resolve real GitHub issues drawn from popular open-source repositories, requiring the agent to localize the relevant files, generate a patch, and pass an associated test suite, providing an objective, executable measure of success rather than relying on human judgment of code quality. GAIA (General AI Assistants) benchmark similarly tests multi-step reasoning combined with tool use, including web search and file parsing, on questions that are simple for humans equipped with the same tools but challenging for a language model to answer directly from parametric knowledge alone.

Despite these advances, benchmark coverage remains skewed toward domains with easily verifiable end-states, such as code correctness or structured web-form completion, while open-ended domains such as multi-session tutoring, exploratory research synthesis, or clinical triage support lack comparably rigorous, automatically verifiable success criteria. This gap between benchmark availability and real-world deployment domains represents a persistent obstacle to systematically comparing agent architectures on the tasks that matter most in practice.

A further complication in agentic evaluation is the distinction between task success rate and process quality: two agents may achieve equivalent final-task success rates while differing substantially in efficiency, measured by the number of tool calls or tokens consumed, and in the interpretability of their intermediate reasoning trace. Comprehensive evaluation protocols increasingly report both outcome-based and process-based metrics to capture this distinction.

XI. CONCLUSION

Agentic AI represents a substantial evolution beyond single-turn language modeling, enabling autonomous systems capable of extended planning, persistent memory, external tool use, and multi-agent collaboration. This review has surveyed the core technical pillars underlying modern agentic systems, examined leading orchestration frameworks including LangGraph, CrewAI, and AutoGen, and discussed emerging applications in healthcare, finance, and education alongside the domain-specific risks each raises. While substantial engineering and research progress has been achieved, open challenges around error propagation, security, coordination overhead, and long-horizon evaluation must be addressed before agentic AI systems can be deployed with full confidence in high-stakes domains. Continued interdisciplinary research spanning systems engineering, security, and human-computer interaction will be essential to realize the full promise of autonomous AI agents.

REFERENCES

[1] S. Yao et al., "ReAct: synergizing reasoning and acting in language models," in Proc. Int. Conf. Learning Representations (ICLR), 2023.

[2] S. Yao et al., "Tree of thoughts: deliberate problem solving with large language models," in Proc. Neural Information Processing Systems (NeurIPS), 2023.

[3] N. Shinn et al., "Reflexion: language agents with verbal reinforcement learning," in Proc. Neural Information Processing Systems (NeurIPS), 2023.

[4] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in Proc. Neural Information Processing Systems (NeurIPS), 2022.

[5] T. Schick et al., "Toolformer: language models can teach themselves to use tools," in Proc. Neural Information Processing Systems (NeurIPS), 2023.

[6] Q. Wu et al., "AutoGen: enabling next-gen LLM applications via multi-agent conversation," arXiv preprint arXiv:2308.08155, 2023.

[7] J. Fourney et al., "CrewAI: a framework for orchestrating role-playing autonomous AI agents," CrewAI Documentation, 2024.

[8] H. Chase, "LangGraph: building stateful, multi-actor applications with LLMs," LangChain Documentation, 2024.

[9] T. R. Sumers et al., "Cognitive architectures for language agents," Transactions on Machine Learning Research, 2024.

[10] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[11] R. E. Fikes and N. J. Nilsson, "STRIPS: a new approach to the application of theorem proving to problem solving," Artificial Intelligence, vol. 2, no. 3-4, pp. 189-208, 1971.

[12] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. Neural Information Processing Systems (NeurIPS), 2020.

[13] K. Cobbe et al., "Training verifiers to solve math word problems," arXiv preprint arXiv:2110.14168, 2021.

[14] L. Wang et al., "A survey on large language model based autonomous agents," Frontiers of Computer Science, vol. 18, no. 6, 2024.

[15] Z. Xi et al., "The rise and potential of large language model based agents: a survey," Science China Information Sciences, vol. 68, 2025.

[16] N. Carlini et al., "Are aligned neural networks adversarially aligned?," in Proc. Neural Information Processing Systems (NeurIPS), 2023.

[17] K. Greshake et al., "Not what you've signed up for: compromising real-world LLM-integrated applications with indirect prompt injection," in Proc. ACM Workshop on Artificial Intelligence and Security, 2023.

[18] OpenAI, "Function calling and structured outputs," OpenAI API Documentation, 2024.

[19] Anthropic, "Tool use with Claude," Anthropic API Documentation, 2024.

[20] Y. Shen et al., "HuggingGPT: solving AI tasks with ChatGPT and its friends in Hugging Face," in Proc. Neural Information Processing Systems (NeurIPS), 2023.

[21] S. Zhou et al., "WebArena: a realistic web environment for building autonomous agents," in Proc. Int. Conf. Learning Representations (ICLR), 2024.

[22] C. E. Jimenez et al., "SWE-bench: can language models resolve real-world GitHub issues?," in Proc. Int. Conf. Learning Representations (ICLR), 2024.

[23] G. Mialon et al., "GAIA: a benchmark for general AI assistants," in Proc. Int. Conf. Learning Representations (ICLR), 2024.