International Journal of Innovative Research in Engineering and Management (IJIREM)
ISSN (Online): 2350-0557, Volume 13, Issue 2, April 2026
DOI: https://doi.org/10.55524/ijirem.2026.13.2.23
Article ID IJIRE-1446, Pages 182-188
www.ijirem.org

Hierarchical Agentic AI Framework for Autonomous Task Planning using Large Language Model Reasoning

Nagaraju Tanguturi[1], and Dr. R. Praveen Kumar[2]

[1] M. Tech Scholar, Department of Computer Science & Engineering, Chaitanya Deemed to be University, Hyderabad, India
[2] Associate Professor, Department of Computer Science & Engineering, Chaitanya Deemed to be University, Hyderabad, India

Correspondence should be addressed to Nagaraju Tanguturi; raju.mca999@gmail.com

Received:2 April 2026; Revised: 16 April 2026; Accepted: 30 April 2026

ABSTRACT- Large Language Models (LLMs) have brought major improvements in natural language understanding, reasoning, and intelligent content generation. Although these models perform well in many applications, most existing AI systems still face difficulties while handling complex multi-step tasks that require planning, coordination, and adaptability. The standard single-agent systems do not work for tasks with multiple dependencies, changing objectives, or long execution workflows.

Hierarchical Agentic AI Framework For Autonomous Task Planning With Large Language Model Reasoning This paper The proposed framework is structured as a hierarchical multi-agent architecture comprising Supervisor Agent, Planning Agent, specialized execution agents and validation layer. The framework aims at analyzing user goals, breaking them down into smaller executable subtasks, and coordinating the execution of the tasks using reasoning-driven workflows.

The system combines chain-of-thought reasoning, contextual memory and intelligent task decomposition to improve the quality of planning and the accuracy of execution. Experimental evaluation demonstrates the superiority of the proposed framework in task completion accuracy, planning efficiency and adaptability, compared to traditional single agent and flat multi-agent architectures. Moreover, the framework shows improved coordination and reduced redundant execution in dynamic workflow settings.

The results show that hierarchical agentic architectures can deliver scalable and intelligent solutions for autonomous workflow execution in enterprise automation, research assistance and AI-driven decision support systems.

KEYWORDS: Agentic AI, Large Language Models, Multi-Agent Systems, Autonomous Task Planning, Hierarchical AI, Workflow Automation, Chain-of-Thought Reasoning, Contextual Memory.

I. INTRODUCTION

In recent years, Artificial Intelligence has evolved rapidly with the development of Large Language Models that can perform advanced reasoning, understand natural language, generate code and make intelligent decisions. Models such as GPT, Llama, Gemini, and Mistral have shown strong performance in many domains and are an important part of modern intelligent systems.

These models are very powerful, but the vast majority of AI applications still employ single-agent architectures. These systems can perform well on individual tasks, but often break down when the task becomes more complex and requires multiple stages of execution, long-term reasoning, contextual awareness, and adaptive planning. These limitations diminish the effectiveness of AI systems in enterprise automation, research workflows, intelligent assistants, and real-time decision-making applications. To address these challenges, researchers have investigated multi-agent AI systems in which multiple specialized agents collaborate to perform tasks. Frameworks like CrewAI, AutoGen and LangChain Agents have brought collaborative workflows with the help of LLM-powered agents. However, many of these systems employ flat coordination approaches where all the agents operate at the same level. Therefore, these systems may suffer from inefficient communication, redundant execution and poor coordination in dealing with complex workflows. Hierarchical agentic architectures offer a more structured solution by introducing supervisory coordination between planning and execution agents. In such systems higher level agents deal with planning, delegation and monitoring; while lower-level agents deal with specialized subtasks. This improves workflow management, task prioritization and execution consistency.

This research proposes a Hierarchical Agentic AI Framework for Autonomous Task Planning using Large Language Model Reasoning. The framework combines a Supervisor Agent, Planning Agent, specialized execution agents, contextual memory, and chain-of-thought reasoning to enable intelligent autonomous task execution. The proposed system is designed to analyze user objectives, generate execution strategies, decompose tasks dynamically, and coordinate specialized agents efficiently.

The main objective of this research is to design a scalable and intelligent autonomous planning framework capable of improving execution quality and reducing workflow inefficiencies in AI-driven environments.

This research makes the following contributions:

- A hierarchical agentic architecture integrating supervisory coordination and specialized execution agents.
- Dynamic task decomposition using Large Language Model reasoning.
- Integration of chain-of-thought reasoning for improved planning and execution sequencing.
- Contextual memory-assisted coordination to improve communication between agents.
- Comparative evaluation against single-agent and flat multi-agent systems.

The remainder of this paper is organized as follows. Section 2 discusses related work and literature review. Section 3 presents the proposed methodology and architecture. Section 4 explains implementation details and experimental evaluation. Section 5 discusses results and performance analysis, while Section 6 concludes the paper and outlines future research directions.

II. LITERATURE SURVEY

The explosion of Large Language Models (LLMs) has triggered a paradigm shift from static text generation to autonomous, multi-step reasoning and decision making. Wei et al. [1], demonstrated that eliciting intermediate reasoning steps dramatically improves LLM performance on arithmetic, symbolic, and commonsense tasks—establishing a foundation for agentic planning.

Subsequent work by Kojima et al. [2] extended this to zero-shot settings, showing that a simple "Let's think step by step" elicitation suffices to unlock latent reasoning without few-shot exemplars. Yao et al. [3] generalised CoT into a Tree of Thoughts (ToT) framework, enabling deliberate search over branching reasoning paths with look-ahead and backtracking—capabilities directly analogous to classical symbolic planners. Together, these techniques supply the reasoning substrate upon which hierarchical agent stacks are constructed.

A. Autonomous Agent Architectures and Hierarchical Planning

Wang et al. [4] surveyed over 150 studies on LLM-based autonomous agents, decomposing them into perception, cognition, and action modules, and identifying persistent challenges in long-horizon planning and memory management. The ReAct framework of Yao et al. [5] operationalised autonomy by interleaving verbal reasoning traces with concrete environment actions in a single LLM output stream, reducing hallucination while enabling dynamic replanning from tool observations. Classical hierarchical task networks (HTNs), formalised by Erol et al. [6], partition goal decomposition into abstract methods and primitive operations, offering correctness guarantees absent in neural planners; their expressivity and completeness results motivate the structural layer of the proposed framework. HuggingGPT (Shen et al. [7]) embodied this orchestration pattern by using ChatGPT as a high-level controller that dispatches subtasks to specialised domain models—concretely demonstrating the manager–worker abstraction central to hierarchical agentic design.

B. Multi-Agent Coordination

Park et al. [8] constructed a society of 25 generative agents whose emergent coordination—driven by natural-language memory streams and periodic reflection—validated decentralised multi-agent LLM systems at social scale. Wu et al. [9] formalized multi-agent conversation as a programming model in AutoGen, which supports role-differentiated agents (planner, coder, critic, executor) with both autonomous and human-in-the-loop modes. Experiments on coding and reasoning benchmarks confirmed consistent gains over single-agent baselines. MetaGPT (Hong et al. [10]) introduced software-engineering role structures (product manager, architect, engineer, QA) and mandated communication through standardized document artifacts that greatly reduced inter-agent hallucination and increased code generation quality. The studies taken together demonstrate that it is the structured communication protocols, and not the multiplicity of agents itself, that is the decisive enabler of scalable hierarchical coordination.

C. Tool-Augmented and Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG), introduced by Lewis et al. [11], grounds LLM outputs in dynamically retrieved, factually current documents, substantially reducing hallucination on knowledge-intensive tasks and providing the factual memory layer that agentic frameworks require. Schick et al. [12] demonstrated in Toolformer that LLMs can learn, via self-supervised training, when and how to invoke external APIs—calculators, search engines, translation services—within the generation stream, generalising grounding from retrieval to arbitrary callable tools. ToolLLM (Qin et al. [13]) extended tool use to over 16,000 real-world APIs by fine-tuning LLaMA with a depth-first search with error-backtracking (DFSDT) decision strategy, showing that open-source models can rival GPT-4 on tool-use benchmarks after targeted training. These three lines of work supply the action execution and knowledge grounding components that distinguish robust agentic pipelines from vanilla LLM inference.

D. Research Gaps

Despite recent progress, several challenges remain in current agentic AI systems:

- Most existing systems rely on flat coordination mechanisms.
- Dynamic task decomposition is still limited in many architectures.
- Inter-agent communication often becomes inefficient in long workflows.
- Existing systems lack effective contextual memory integration.
- Many frameworks struggle with adaptive planning and execution consistency.

The proposed research addresses these limitations by introducing a hierarchical framework integrating planning intelligence, contextual memory, and reasoning-driven coordination.

III. PROPOSED METHODOLOGY

The proposed Hierarchical Agentic AI Framework is designed to autonomously analyze user objectives, generate

execution strategies, decompose tasks dynamically, and coordinate specialized agents using Large Language Model reasoning.

The framework consists of multiple intelligent agents organized into supervisory and execution layers.

A. System Architecture

The architecture consists of the following major components (See the Figure 1).

i) User Interaction Layer

This layer receives user requests through a web-based interface and forwards them to the Supervisor Agent. Users may provide tasks such as report generation, research analysis, workflow automation, or content summarization.

ii) Supervisor Agent

The Supervisor Agent acts as the central coordinator of the framework. Its responsibilities include:

- Understanding user objectives
- Managing workflow execution
- Monitoring agent communication
- Handling task dependencies
- Coordinating final output generation

The Supervisor Agent maintains overall contextual awareness throughout execution.

iii) Planning Agent

The Planning Agent is the main reasoning component of the system. It uses Large Language Model reasoning and chain-of-thought prompting to:

- Analyze task complexity
- Generate execution plans
- Decompose tasks into subtasks
- Prioritize execution order
- Allocate tasks dynamically

This module enables intelligent planning and adaptive workflow management.

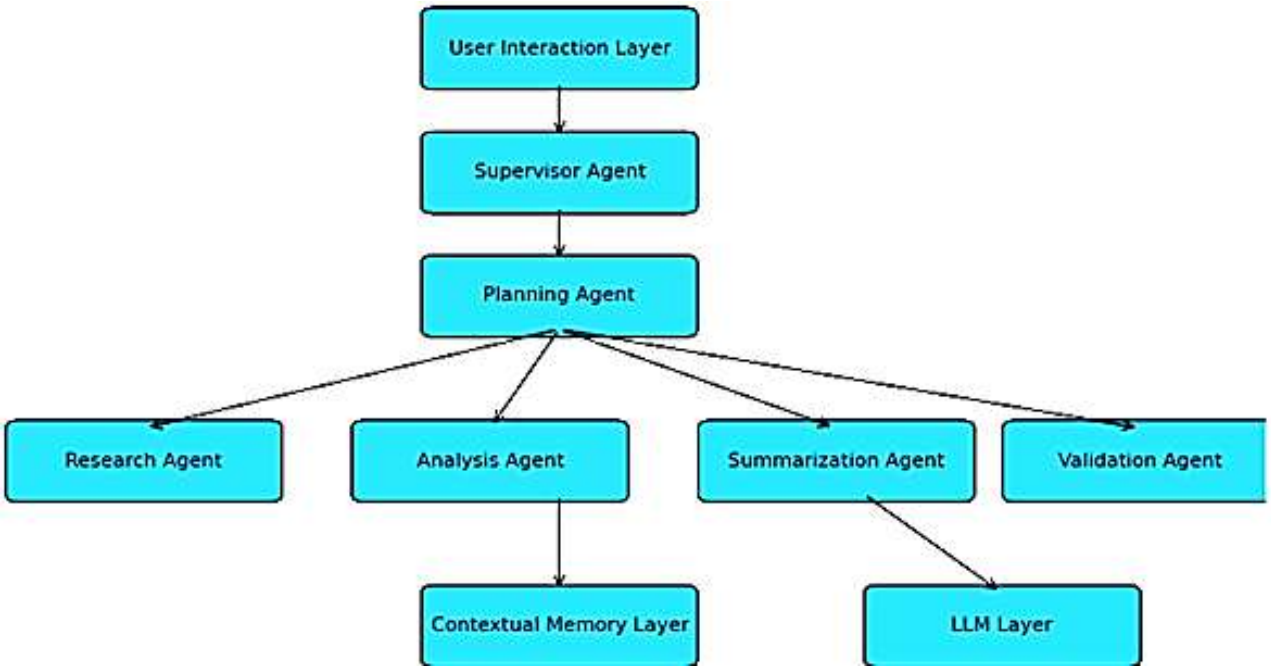

Figure 1: System Architecture

iv) Specialized Execution Agents

The framework includes multiple execution agents responsible for domain-specific tasks.

- ***Research Agent-*** Responsible for information retrieval, document analysis, and semantic search.
- ***Analysis Agent-*** Processes retrieved data and performs reasoning and analytical operations.
- ***Summarization Agent-*** Generates concise summaries and structured outputs.
- ***Validation Agent-*** Validates execution quality, checks consistency, and minimizes hallucination errors.

v) Contextual Memory Layer

The memory layer stores:

- Execution history
- Intermediate outputs
- Context embeddings
- Task dependencies

A vector database maintains semantic context and improves inter-agent coordination.

vi) Large Language Model Layer

This layer provides reasoning and natural language generation capabilities. The framework supports both local and cloud-based LLMs.

B. Dynamic Task Decomposition

Dynamic task decomposition is one of the important features of the proposed framework. Instead of generating outputs directly, the system first analyzes the objective and divides it into smaller executable subtasks.

For example, a request such as:

"Generate a market analysis report for electric vehicles"

is decomposed into the following subtasks:

- Collect market statistics
- Identify major companies
- Analyze growth trends

- Generate comparative insights
- Produce the final report

This approach improves execution efficiency and enables distributed processing among specialized agents.

C. Chain-of-Thought Reasoning

The framework uses chain-of-thought reasoning to improve planning quality and logical consistency. Instead of producing direct outputs, the Planning Agent generates intermediate reasoning steps before execution (See the Figure 2). The workflow follows:

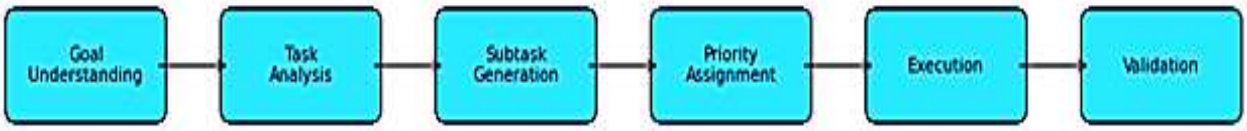

Figure 2: Workflow Diagram

This reasoning mechanism improves contextual understanding and reduces execution errors.

IV. IMPLEMENTATION AND EXPERIMENTAL EVALUATION

A. Implementation Environment

The proposed framework was implemented using Python along with modern agentic AI frameworks and Large Language Models.

Table 1: Implementation Environment of the Proposed Hierarchical Agentic AI Framework

Component	Technology
Programming Language	Python
Agent Framework	CrewAI
LLM Framework	LangChain
Vector Database	FAISS
User Interface	Streamlit
Local Runtime	Ollama
LLM Models	Llama 3, Mistral

Table 1 presents the implementation environment used for developing the proposed Hierarchical Agentic AI Framework for Autonomous Task Planning. The system was implemented using Python as the primary programming language due to its extensive support for Artificial Intelligence and machine learning applications. CrewAI was utilized as the agent orchestration framework to enable coordination among multiple intelligent agents, while LangChain was employed for integrating Large Language Model reasoning and workflow management functionalities. FAISS was used as the vector database for efficient semantic search and contextual memory storage. The user interface was developed using Streamlit to provide an interactive and user-friendly environment for task execution and monitoring. Ollama served as the local runtime environment for executing Large Language Models locally, and advanced LLMs such as Llama 3 and Mistral were integrated to support reasoning, planning, and intelligent task execution capabilities within the proposed framework.

B. Experimental Setup

The framework was evaluated using multiple complex task scenarios including:

- Research report generation
- Multi-step workflow automation
- Information analysis
- Task prioritization
- Context-aware summarization

The proposed framework was compared with:

- Single-Agent Systems
- Flat Multi-Agent Systems
- Proposed Hierarchical Agentic Framework

C. Evaluation Metrics

The following performance metrics were used during evaluation.

- **Task Completion Accuracy-** Measures successful execution of generated subtasks.
- **Planning Efficiency-** Measures logical sequencing and workflow quality.
- **Execution Latency-** Measures total workflow execution time.
- **Coordination Efficiency-** Measures communication and synchronization between agents.
- **Adaptability-** Measures the ability to handle dynamic workflow changes.

D. Experimental Results

Table 2: Performance Comparison

System Type	Task Accuracy	Planning Efficiency	Adaptability
Single-Agent System	71%	65%	58%
Flat Multi-Agent System	84%	79%	74%
Proposed Hierarchical Framework	93%	91%	89%

Table 2 and Figure 3 present the comparative performance evaluation of different AI system architectures based on task accuracy, planning efficiency, and adaptability. These results demonstrate that the hierarchical coordination mechanism, combined with intelligent task decomposition, contextual memory, and chain-of-thought reasoning, significantly improves workflow management, execution quality, and adaptive decision-making capabilities in autonomous agentic AI systems.

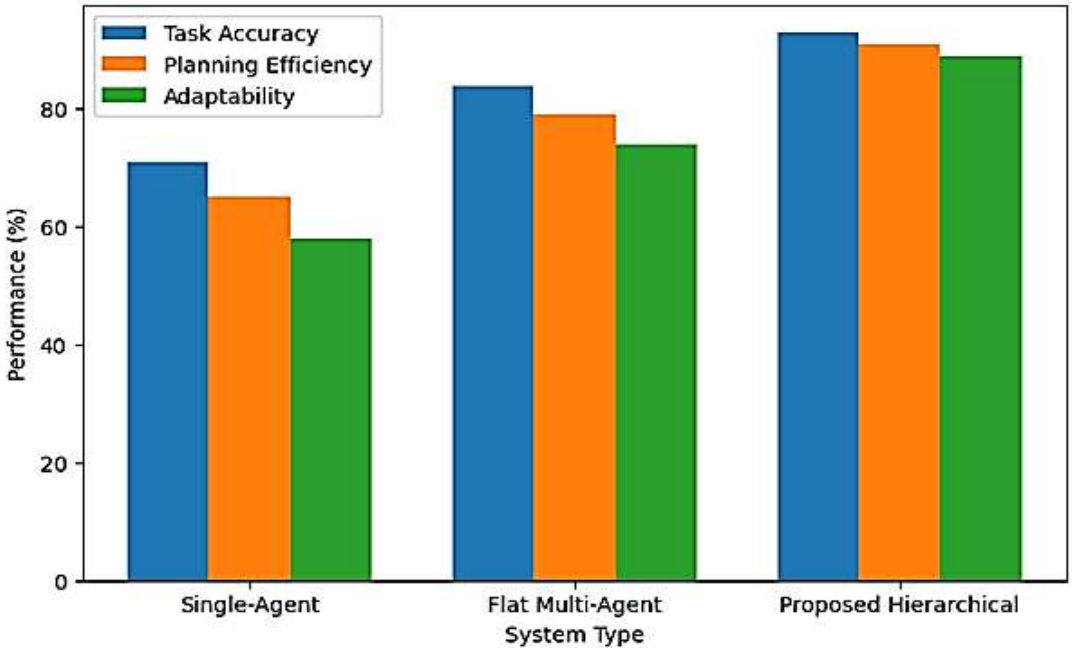

Figure 3: Comparative Performance Analysis

V. RESULTS AND DISCUSSION

The experimental evaluation shows that the proposed Hierarchical Agentic AI Framework improves autonomous task planning, workflow coordination and execution quality significantly compared to traditional single-agent and flat multi-agent systems. We combined supervisory coordination, context memory, dynamic task decomposition, and chain-of-thought reasoning for more efficient and adaptive workflow execution.

The Planning Agent successfully generated structured execution workflows using chain-of-thought reasoning, which improved logical task sequencing and dependency management. By decomposing complex objectives into smaller executable subtasks, the framework reduced execution ambiguity and enabled specialized agents to process domain-specific tasks more efficiently. The Supervisor Agent further enhanced coordination among execution agents by monitoring communication flow, resolving task dependencies, and minimizing redundant processing operations.

The contextual memory layer played a significant role in maintaining information continuity across long execution workflows. By storing intermediate outputs, contextual embeddings, and execution history in the vector database, the system improved consistency in generated responses and reduced hallucination errors during multi-step reasoning tasks. This capability was especially beneficial in research analysis and workflow automation scenarios requiring long-context understanding.

Compared to traditional single-agent systems, the proposed framework demonstrated:

- Higher task completion accuracy
- Better reasoning quality
- Improved adaptability
- Efficient workflow management
- Reduced redundant execution

The performance comparison presented in Table 2.

A. Latency Analysis

Execution latency was analyzed to evaluate the responsiveness of the proposed framework during complex multi-step workflows. The hierarchical framework demonstrated lower average execution latency compared to flat multi-agent systems because tasks were organized hierarchically and delegated efficiently among specialized agents. The Planning Agent reduced unnecessary execution cycles by generating optimized task sequences before workflow initiation.

Although the framework introduced additional supervisory coordination layers, the reduction in redundant processing and improved task prioritization compensated for this overhead. The use of contextual memory further minimized repeated reasoning operations by reusing previously generated embeddings and intermediate outputs. As a result, the framework maintained stable response times even for workflows involving multiple reasoning stages and interdependent subtasks.

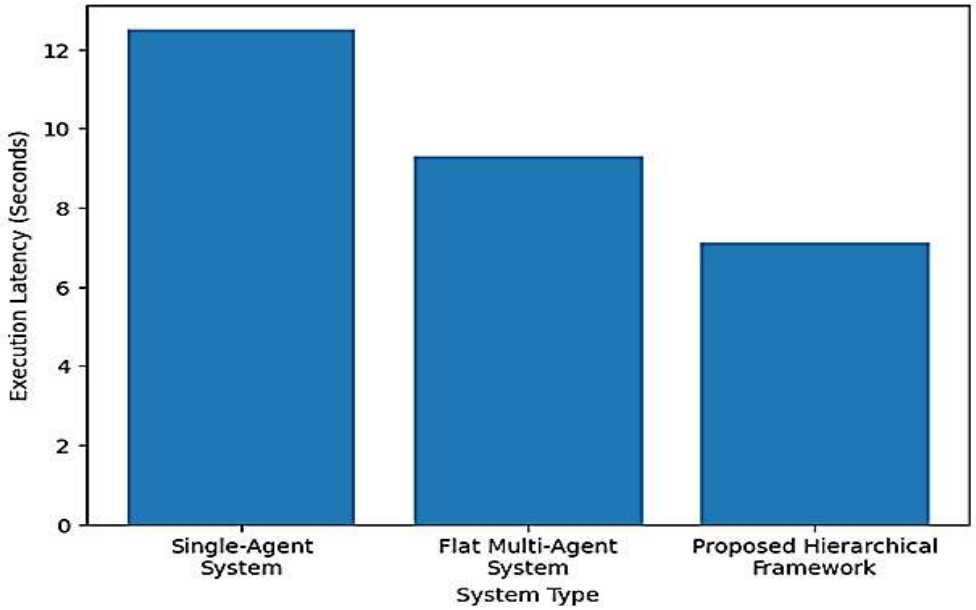

Figure 4: Execution latency comparison of different AI framework

Figure 4 illustrates the execution latency comparison among the Single-Agent System, Flat Multi-Agent System, and the Proposed Hierarchical Framework. The proposed hierarchical framework achieved the lowest execution latency of approximately 7.1 seconds compared to 9.3 seconds for the flat multi-agent system and 12.5 seconds for the single-agent system. The results indicate that hierarchical coordination, optimized task decomposition, and contextual memory integration significantly improve workflow responsiveness and reduce execution time in complex multi-step AI tasks.

B. Scalability Evaluation

Scalability evaluation showed that the proposed architecture effectively handled increasing workflow complexity and agent interactions. The hierarchical coordination mechanism enabled efficient distribution of subtasks among specialized execution agents without significantly affecting planning quality or execution consistency (See the Figure 1) .

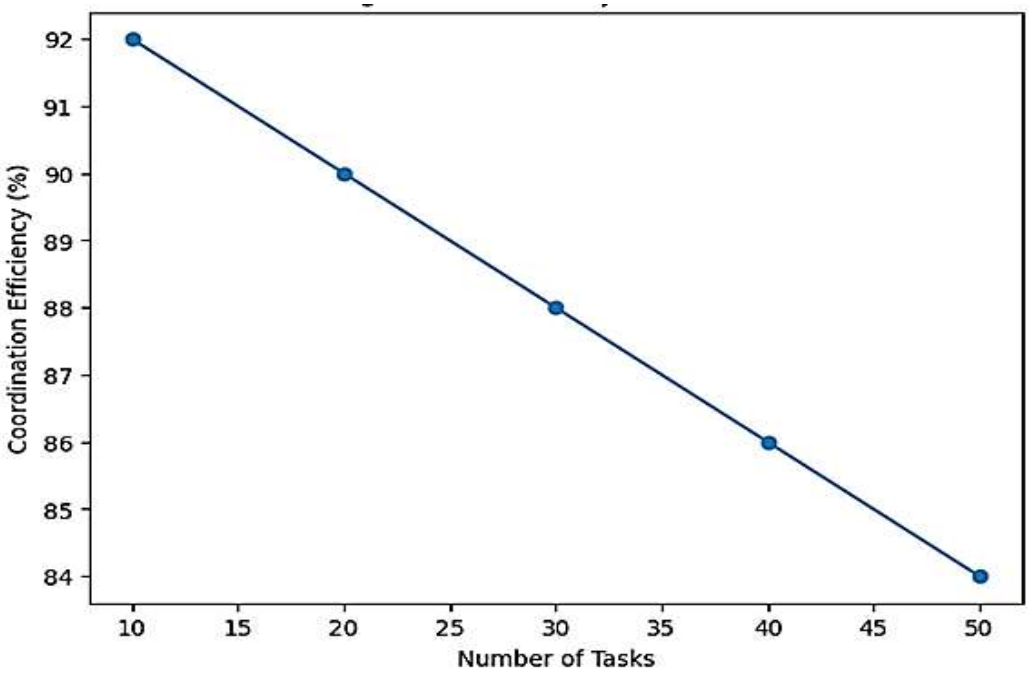

Figure 5: Scalability Evaluation of the Proposed Framework

The Figure 5(graph) illustrates the scalability performance of the proposed Hierarchical Agentic AI Framework by analyzing the relationship between the number of tasks and coordination efficiency. The X-axis represents the number of tasks processed by the system, while the Y-axis represents coordination efficiency measured in percentage. The results show that as the number of tasks increases from 10 to 50, the coordination efficiency gradually decreases

from 92% to 84%. Although a slight reduction in efficiency is observed with increasing workload, the framework continues to maintain relatively high coordination performance even under larger task volumes. This indicates that the hierarchical coordination mechanism effectively manages communication, task synchronization, and workflow execution across multiple agents.

The gradual decline in efficiency is expected because larger workflows require additional inter-agent communication, memory management, and execution monitoring. However, compared to traditional flat multi-agent architectures, the proposed framework demonstrates better scalability and controlled performance degradation due to its centralized supervisory coordination and structured task decomposition strategy.

Overall, the graph validates that the proposed hierarchical framework can efficiently scale to handle increasing workflow complexity while maintaining stable coordination and execution quality in autonomous AI-driven environments.

VI. CONCLUSION AND FUTURE WORK

This research presented a Hierarchical Agentic AI Framework for Autonomous Task Planning using Large Language Model Reasoning. This framework enables intelligent autonomous workflow execution through supervisory coordination, dynamic task decomposition, contextual memory and chain-of-thought reasoning. Experimental evaluation has shown that hierarchical multi-agent coordination improves planning efficiency, execution quality and adaptability compared to classical single-agent and flat multi-agent systems.

The proposed framework was capable of managing complex multi-step workflows via intelligent reasoning and coordinated execution strategies. The architecture is suitable for enterprise automation, research support, workflow orchestration and AI-based decision support systems.

Future work will focus on:

- Self-improving agent architectures
- Reinforcement learning-based planning optimization
- Real-time collaborative agent communication
- Multimodal reasoning integration
- Distributed cloud-scale agentic systems

The integration of adaptive learning mechanisms and advanced reasoning strategies is expected to further improve the intelligence and scalability of future agentic AI systems.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 24824–24837, 2022. Available from: https://arxiv.org/abs/2201.11903

[2] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 22199–22213, 2022. Available from:: https://proceedings.neurips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html

[3] S. Yao et al., "Tree of thoughts: Deliberate problem solving with large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 11809–11822, 2023. Available from: https://papers.nips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html

[4] L. Wang et al., "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, no. 6, p. 186345, 2024. Available from: https://doi.org/10.1007/s11704-024-40231-1

[5] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *Proc. 11th Int. Conf. Learn. Representations (ICLR)*, 2023. Available from: https://arxiv.org/abs/2210.03629

[6] K. Erol, J. Hendler, and D. S. Nau, "HTN planning: Complexity and expressivity," in *Proc. 12th Nat. Conf. Artificial Intelligence (AAAI)*, vol. 94, pp. 1123–1128, 1994. Available from: https://cdn.aaai.org/AAAI/1994/AAAI94-173.pdf

[7] Y. Shen et al., "HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 38154–38180, 2023. Available from: https://arxiv.org/abs/2303.17580

[8] J. S. Park et al., "Generative agents: Interactive simulacra of human behavior," in *Proc. 36th Annu. ACM Symp. User Interface Softw. Technol. (UIST '23)*, 2023, pp. 1–22. Available from: https://doi.org/10.1145/3586183.3606763

[9] Q. Wu et al., "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," *arXiv preprint arXiv:2308.08155*, 2023. Available from: https://arxiv.org/abs/2308.08155

[10] S. Hong et al., "MetaGPT: Meta programming for a multi-agent collaborative framework," in *Proc. 12th Int. Conf. Learn. Representations (ICLR)*, 2024. Available from: https://arxiv.org/abs/2308.00352

[11] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020. Available from: https://arxiv.org/abs/2005.11401

[12] T. Schick et al., "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 68539–68551, 2023. Available from: https://arxiv.org/abs/2302.04761

[13] Y. Qin et al., "ToolLLM: Facilitating large language models to master 16000+ real-world APIs," in *Proc. 12th Int. Conf. Learn. Representations (ICLR)*, 2024. Available from: https://arxiv.org/abs/2307.16789