International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A.A.A.R.M.I.S: Agent Auto a Rather Moderately Intelligent System

K Balamuralikrishna[a], Ganesha D[b] , Dheemanth R[c], Gowrav K[d], Dr. S Prabhanjan[e]

[a,b,c,d] Department Of Computer Science and Engineering, Jyothy Institute Of Technology, Bengaluru, India
[e] Guide, Head of Department, Computer Science and Engineering, Jyothy Institute Of Technology, Bengaluru, India

A B S T R A C T :

The rapid evolution of Large Language Models (LLMs) has transitioned Artificial Intelligence from passive information retrieval to active task execution. However, current digital assistants remain limited by rigid command sets, lack of contextual awareness, and inability to perform deep Operating System (OS) level operations. To address these fragmentation issues, this paper presents A.A.A.R.M.I.S (Agent Auto a Rather Moderately Intelligent System), a comprehensive survey and implementation framework for an autonomous agentic system. The study explores the integration of Model Context Protocol (MCP) with localized LLM reasoning to enable multi-step, crossplatform workflows. The proposed system utilizes a hybrid architecture combining Electron, React, and high-performance Rust binaries for low-level system hooking. By analyzing state-ofthe-art agentic workflows, reinforcement learning, and retrievalaugmented generation (RAG), this paper outlines a unified approach to creating an "always-listening" assistant capable of planning, executing, and verifying complex tasks across Windows and Linux environments.

Keywords: Agentic AI, Model Context Protocol, Desktop Automation, Rust, Electron, HumanComputer Interaction.

1. INTRODUCTION :

The modern digital landscape is characterized by complexity and fragmentation. Users frequently navigate disparate applications, Operating Systems (OS), and external devices to complete routine tasks. While current digital assistants (e.g., Siri, Alexa) have successfully simplified basic interactions, their utility is fundamentally limited by rigid, predefined command sets and a lack of contextual awareness across platforms. The proliferation of powerful, locally deployable Large Language Models (LLMs) and the Model Context Protocol (MCP) have now made it possible to design truly autonomous agents capable of advanced reasoning and realworld execution.

A.A.A.R.M.I.S. (Agent Auto a Rather Moderately Intelligent System) was conceived to meet this demand. It integrates multi-modal input (voice and text), agentic planning, and platform-specific adapters to create a unified system capable of autonomously planning, executing, and verifying complex workflows. Traditional automation tools like Robotic Process Automation (RPA) are often hardcoded and applicationspecific, lacking real-time adaptability. There is a pressing need for a modular AI assistant that combines agentic reasoning with feedback loops to handle dynamic errors and changing environments.

This paper presents a plenary survey on the technologies underpinning A.A.A.R.M.I.S, reviewing up-to-date advancements in Natural Language Processing (NLP), Reinforcement Learning, and Agentic Workflows. Furthermore, it details the implementation of a prototype system that bridges the gap between high-level AI reasoning and low-level OS interaction.

2. BACKGROUND AND RELATED WORK

A. Background

The modernization of desktop automation has been propelled by the integration of AI and OS-level scripting. However, despite the progress of LLMs, existing systems often face significant constraints—specifically the "isolation" of the AI from the actual operating environment. To overcome these challenges, researchers are now steering toward "Agentic AI," systems designed for complex, long-term goals with minimal human intervention.

B. Related Work

AI has proven to be a game changer in automation. However, distinct areas of research must converge to create a truly autonomous desktop agent.

1) *Agentic AI and Workflows:* Acharya et al. [1] provide a comprehensive survey of Agentic AI, defining it through characteristics such as adaptability, decision-making, and selfsufficiency. This foundation is critical for moving beyond simple chatbots to systems that act. Complementing this, Singh et al. [2] explore "Agentic Workflows" in LLMs, highlighting patterns like Reflection, Planning, and Tool Utilization. Their findings suggest that iterative engagement significantly improves outcomes in code generation and complex problem solving.

2) *Multimodal Understanding and RAG:* To function effectively, an agent must understand both text and visual context. Radford et al. [8] introduced CLIP, enabling zero-shot transfer for visual concepts, which is essential for UI element identification. Simultaneously, Meta AI Research [7] highlights the importance of Retrieval-Augmented Generation (RAG) to mitigate hallucinations and provide dynamic knowledge retrieval.

3. SYSTEM ARCHITECTURE

The proposed system architecture is structured to maximize performance and security across platforms. It utilizes a threetiered architecture built upon the Electron framework, ensuring a separation of concerns between the user interface and system-level operations.

A. Architectural Layers

The system consists of three distinct layers:

1) The Client Layer (UI/Renderer): Built with React and TypeScript, managing the user interface, conversation display, and real-time progress visualization.
2) The Orchestration Layer (Electron Main Process): Acts as the "brain," coordinating inputs, managing state, handling LLM routing, and supervising the MCP Agent lifecycle.
3) The External Services Layer: Includes third-party AI APIs (OpenAI, Groq, Gemini) and remote MCP Servers for task execution.

B. Low-Level System Integration (Rust)

A unique feature of this implementation is the use of a Rust binary for operations requiring global OS access, which is often unreliable in pure Node.js environments.

- Global Hotkey Listening: Leveraging the rdev crate to monitor keyboard events outside the application focus.
- Text Injection: Using the enigo crate to simulate programmatic typing, allowing the agent to inject output directly into active applications.

4. IMPLEMENTATION METHODOLOGY

The core methodology follows a "Think-Act-Observe" loop enabled by the Model Context Protocol (MCP).

A. Input Acquisition

The methodology begins with input capture decoupled from the main Node.js process for stability. A compiled Rust binary executes in "listen" mode. Upon detecting a configured hotkey event, it serializes the data to JSON and signals the Electron Main Process to initiate the agent flow. Voice input is transcribed using the Whisper model.

B. AI Processing and Decision Flow

The Orchestration Layer utilizes an LLM to decompose the user request. The methodology distinguishes between two modes:

1. Dictation Mode: Focused on accurate Speech-to-Text and direct injection.
2. Agent Mode: The LLM utilizes defined MCP tools (File System, Web Search, etc.). It generates a plan, calls the appropriate tool, and waits for the result.

C. Execution and Feedback

The agent acts as an MCP Client. It executes tools with support for OAuth 2.1 authentication. The output of the tool is fed back into the context window as an observation. The LLM then reflects on this output to determine if the task is complete or if replanning is required. Finally, the result is injected into the user's active application using the Rust-based enigo simulation.

CONCLUSION

The development of A.A.A.R.M.I.S represents a significant step toward bridging the gap between natural language understanding and operating system execution. By integrating Agentic AI workflows with the robust Model Context Protocol, the system demonstrates how digital assistants can evolve from passive chat interfaces to active, autonomous collaborators. Unlike traditional automation scripts, this framework leverages a hybrid Rust-Electron architecture to ensure high-performance input monitoring and secure execution. The findings underscore the potential of combining local system control with cloud-based reasoning to create scalable, reliable, and user-centric automation tools.

REFERENCES

[1] D. B. Acharya, K. Kuppan, and B. Divya, "Agentic AI: Autonomous intelligence for complex goals—A comprehensive survey," *IEEE Access*, vol. 13, pp. 18912–18935, 2025.

[2] A. Singh, A. Ehtesham, S. Kumar, and T. Talaei Khoei, "Enhancing AI systems with agentic workflows patterns in large language model," in *Proc. IEEE World AI IoT Congress (AIIoT)*, 2024, pp. 527–532.

[3] D. Huh and P. Mohapatra, "Multi-agent reinforcement learning: A comprehensive survey," *arXiv preprint arXiv:2312.10256v2*, 2024.

[4] A. Konn, "The intersection of computer science and cybersecurity: Safeguarding devices in the era of cloud-based technology," *ResearchGate*, Dec. 2018.

[5] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," presented at *Int. Conf. Learn. Represent. (ICLR)*, 2016.

[6] Z. McBride Lazri et al., "MAFE: Multi-agent fair environments for decision-making systems," *arXiv preprint arXiv:2502.18534v1*, 2025.

[7] Meta AI Research, "Advancing retrieval-augmented generation (RAG): Innovations, challenges, and the future of AI reasoning," 2024.

[8] A. Radford et al., "Learning transferable visual models from natural language supervision," *arXiv preprint arXiv:2103.00020v1*, 2021.

[9] P. S. Viswanathan, "Agentic AI: A comprehensive framework for autonomous decision-making systems in artificial intelligence," *Int. J. Comput. Eng. Technol. (IJCET)*, vol. 16, no. 1, pp. 862–880, Jan.-Feb. 2025.

[10] M. J. Basha, S. Vijayakumar, J. Jayashankari, A. H. Alawadi, and D. Pulatova, "Advancements in natural language processing for text understanding," in *E3S Web of Conferences*, vol. 399, Art. no. 04031, 2023.