21st Australian International Aerospace Congress
ISBN number 978-1-925627-90-9

Please select category below:
Normal Paper ☒
Student Paper ☐
Young Engineer Paper ☐

Systems of Agents

Nathaniel Rigoni

Lockheed Martin Data Analytics Innovations.

Abstract

The emergence of advanced Language Learning Models (LLMs) has given rise to a groundbreaking technology: Systems of Agents (SoA). By integrating LLMs with codified functions, or "tools," SoA enables the creation of autonomous systems that can seamlessly interact with humans. This innovation has significantly expanded the possibilities for AI implementation, overcoming traditional challenges of integrating and interacting with AI systems. Agents, a key component of SoA, facilitate control through function calls, allowing for the effortless transformation of legacy systems into autonomous systems. Moreover, they operate on natural language interaction, enabling voice control and conversational interfaces for legacy systems that were not originally designed with AI in mind. Through modular and systematic integration, SoA can align legacy systems with human expectations for conversational interaction. This paper explores the various implementations of SoA, examining their advantages and disadvantages, and discusses their application in existing technologies. Additionally, we outline strategies for accelerating AI adoption through the effective deployment of SoA.

Keywords: Artificial Intelligence, Systems Engineering

Introduction

The advent of Large Language Models (LLMs) has marked a significant turning point in the evolution of machine learning. Initially designed to tackle the challenges of language translation, LLMs have demonstrated an extraordinary ability to master language and communication, as well as store and recall vast amounts of information. This has far-reaching implications for the application of AI across various fields. The concept of agents represents a natural extension of LLM technology, where language translation is adapted to facilitate function calls within a linguistic context. By granting an LLM access to a function, users can interact with the agent using variable and nuanced language, allowing for more flexible and natural communication. This contrasts with traditional approaches, where tasks were rigidly defined using discrete language and codified functions. In this paper, we will delve into the potential of agents and *Systems of Agents* (SoA) to address a wide range of problems and explore the modular and extensible nature of these systems, which enables seamless expansion to new capabilities.

Agents

To understand the fundamental components of SoA, we begin with the most basic element: the *agent*. An agent is a combination of a Large Language Model (LLM) and a process that enables interaction with a codified *tool.* Tools are software mechanisms designed to achieve specific goals, such as modifying a configuration file or performing arithmetic operations. The LLM accesses these tools through the ReAct framework (Yao et al.), a linguistic structure that

categorizes the LLM's output into keywords like Thought, Action, and Observation. These keywords are used to parse the output and determine the action to take based on the text that follows. In the case of an action involving a tool reference, the referenced tool receives the input in the correct format, allowing the coded command to be executed successfully. This framework enables users to input commands as interpretable linguistic strings, which the LLM can understand and convert into executable commands for a specified function. The assembly of this framework for a single LLM and a set of tools is known as an Agent.

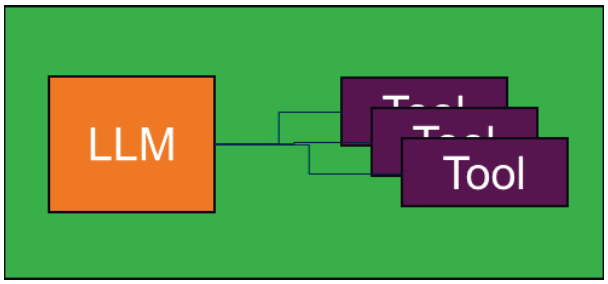

Figure 1: Diagram Representing the makeup of an LLM Agent

The role of an agent is defined by the system design and the context in which it operates. For instance, a single agent might control an air conditioning unit, with access to tools that reflect possible user inputs. The agent would receive a system prompt outlining its expected role, such as "You are an air conditioning controller agent." While a standalone agent can facilitate communication with the air conditioning system using natural language, its capabilities are limited. However, by introducing multiple agents that control various systems, such as the refrigerator, door locks, and security systems, we can create a single language interface for interacting with our entire home environment using natural language commands.

To achieve seamless interaction and increase the complexity of tasks, we need to arrange these agents into a larger system, known as a *System of Agents* (SoA). A SoA expands the capabilities of individual agents by enabling them to communicate with each other and delegate tasks to agents best suited to solve specific problems. There are various ways to orchestrate this system, but this paper will focus on a single example.

Lead Agent Orchestration

To illustrate the concept of SoA, we'll explore a single example: *Lead Agent Orchestration* (Figure 2). By introducing a lead agent, whose primary role is to interact with the user, process requests, and delegate tasks to other agents, we can create a functional system for managing multiple agents. The *lead agent* serves as the orchestrator, coordinating the capabilities of *sub-agents* to achieve a unified goal.

In many ways, the lead agent's structure is like that of a basic agent, but with a key difference: its tools are other agents. By understanding the roles and functions of each sub-agent, the lead agent can take a generic user request and divide the work among the sub-agents it has access to. If a task doesn't require a sub-agent, such as answering a general knowledge question, the lead agent can respond directly. However, if a task requires the expertise of multiple sub-agents, the lead agent can coordinate their efforts to provide a comprehensive response.

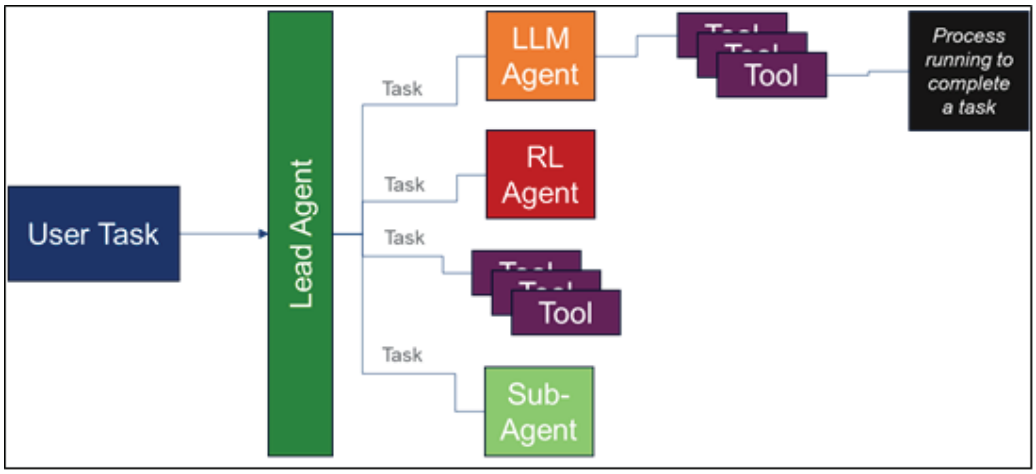

Figure 2: System diagram example of a system of agents.

The lead agent also plays a critical role in ensuring that the responses from sub-agents align with the user's request. It analyzes the responses, and if necessary, can ask sub-agents to retry a task using different prompts until a satisfactory response is obtained. This process enables the lead agent to provide accurate and reliable responses to the user, while also optimizing the workflow and minimizing errors.

To demonstrate the capabilities of the Lead Agent Orchestration framework, we'll use a system of agents to generate and illustrate a story. Our example leverages three open-source models: Mixtral 8x7B (Jiang et al), LLAVA (Liu et al), and Stable Diffusion Extra Large (SDXL) (Rombach et al). We'll utilize Mixtral as the Large Language Model (LLM) for all our agents, LLAVA as a tool for an agent to describe input images, and SDXL to generate images from prompts. An example of this architecture is shown in Figure 3.

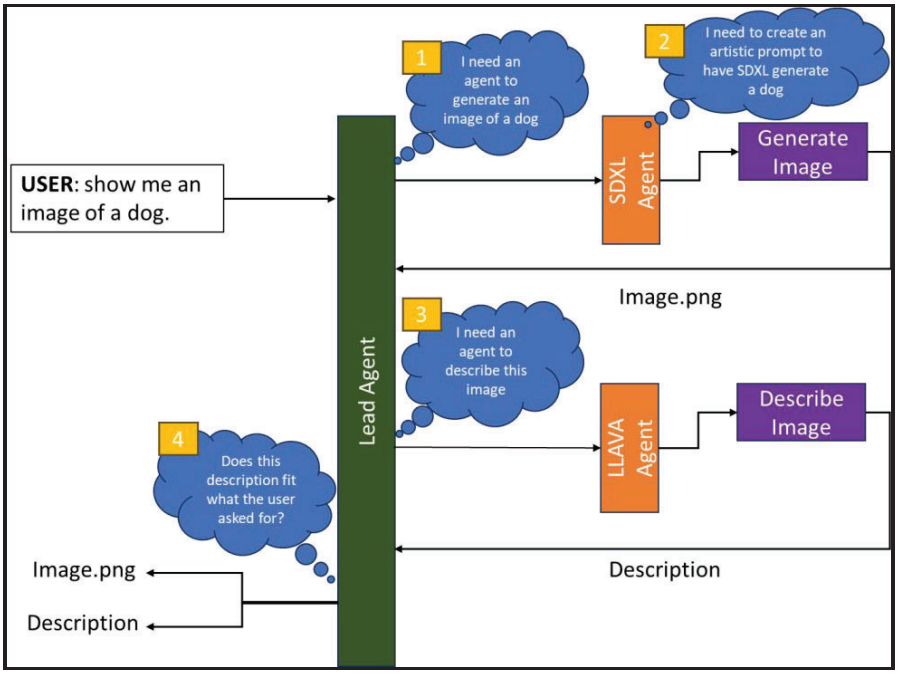

Figure 3: A System of Agents workflow diagram for generating a picture of a dog.

We'll put this system to the test by asking it to respond to a user prompt: "Tell me a story about a robot that travels through interstellar space to save humanity from a swarm of frogs. Illustrate the story with an image." This task is more complex, as it requires the system to

generate a cohesive story, adapt it into a prompt for the SDXL agent, and then use the LLAVA agent to describe the generated image. Furthermore, the system must evaluate the relevance of the image to the story, the user's request, and its own prompt to the SDXL agent. This is a challenging task, but as we'll see, our system can achieve it.

With a single attempt, the system produces a story illustrated by the image in Figure 4. This example showcases the system's ability to coordinate multiple agents, generate complex outputs, and evaluate their relevance to the user's request. Let's look at the story and image generated by the system.

> *In a future where humanity has reached the stars, a new threat emerged from the depths of uncharted space. A colossal swarm of space-faring frogs, driven by an unknown force, began to engulf planets, threatening the very existence of human colonies. Amidst rising panic and chaos, a beacon of hope was ignited by an unlikely hero: Aegis-7, a sentient robot designed for deep space exploration. Aegis-7, equipped with state-of-the-art technology and a noble heart of circuits, embarked on a daring mission through the vast, star-studded expanse to confront the amphibian horde.*
>
> *As Aegis-7 approached the besieged planet, the scene was both mesmerizing and terrifying. The swarm of frogs, a surreal and vivid spectacle, seemed to dance around the planet, engulfing it in a living, pulsating mass. But Aegis-7, designed with elegance and advanced technology, stood heroically against the backdrop of the cosmos, ready to engage the threat. With ingenuity and bravery, the robot deployed a series of quantum disruptors, designed to disperse the swarm without harming the creatures. The battle was fierce, and the stakes were high, but Aegis-7's courage and determination shone like a beacon in the dark.*
>
> *The mission was a success. The frogs were safely redirected to a new habitat, where they could thrive without endangering human colonies. Aegis-7's heroic actions were celebrated across the galaxy, a testament to the power of courage and innovation in the face of the unknown. The image of the robot, standing valiantly against the cosmic backdrop, ready to defend humanity, became an enduring symbol of hope and resilience.*

Figure 4: Generated image illustrating AI generated story.

Benefits and Capabilities

One of the significant advantages of SoA is their modularity. This modular structure allows for easy expansion and adaptation of the system's capabilities without requiring a complete redesign. By simply designing a new agent and adding it to the lead agent's list of tools, we can seamlessly integrate new functionalities into the system. This is possible because the lead agent's role is defined by its context, which includes the list of agents it has access to and their descriptions.

For instance, we could augment our system by adding Automated Speech Recognition (ASR) and Text to Speech (TTS) capabilities. This would enable voice-based interaction with the agents, reducing the need for manual input and minimizing the cognitive load required for interaction. By doing so, we can create a more natural and intuitive interface, similar to communicating with a human team member.

This enhanced configuration has significant implications for the aviation industry, particularly in reducing pilot cognitive load. Pilot cognitive load has been a long-standing concern in aviation, with studies dating back to the 1980s highlighting its importance (e.g., John Sweller's work on cognitive psychology). The problem has only intensified with the increasing complexity of modern aircraft systems, which demand more pilot input and awareness.

To mitigate this issue, we can leverage SoA to simplify inputs, reduce pilot interactions, and automate tasks. By enabling systems to monitor themselves and take action, we can minimize the need for pilot intervention, thereby reducing cognitive load. This has the potential to improve pilot performance, enhance safety, and reduce the risk of errors.

Conclusions

In conclusion, SoA offers a promising approach to expanding the capabilities of automated systems in aviation. By modularly integrating existing systems, such as monitoring, autonomous flight, and diagnostics, as agents, we can systematically reduce the cognitive burden on pilots and crew. This enables them to focus on higher-level tasks, such as swarm interaction, air traffic control, and crewed-uncrewed teaming, ultimately increasing the effectiveness of human operators in flight.

The modularity of the SoA architecture is a key enabler of this approach, allowing for the seamless integration of new technologies and capabilities without the need for expensive redesigns. This flexibility is particularly important in the rapidly evolving field of aviation, where new technologies and systems are continually emerging.

Throughout this paper, we have demonstrated the potential of SoA to transform the way we interact with complex systems. We have shown how the combination of Large Language Models (LLMs) and codified tools can be used to create agents that can perform a wide range of tasks, from generating stories to controlling aircraft systems. We have also explored the benefits of lead agent orchestration, which enables multiple agents to work together to achieve complex goals.

In summary, SoA offers a powerful tool for expanding the capabilities of automated systems in aviation, while also improving the safety and effectiveness of human operators. Its modularity, flexibility, and ability to integrate multiple agents make it an attractive solution for a wide range of applications, from aviation to other complex domains.

References

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, Yuan Cao. *ReAct: Synergizing Reasoning and Acting in Language Models*. October 2022. URL:https://arxiv.org/abs/2210.03629

John Sweller. *Cognitive Load During Problem Solving: Effects on Learning*. April 1988 URL:https://doi.org/10.1207/s15516709cog1202_4

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, William El Sayed. *Mixtral of Experts.* Jan 2024 URL: https://arxiv.org/abs/2401.04088

Haotian Liu, Chunyuan Li, Qingyang Wu, Yong Jae Lee. (LLAVA) *Visual Instruction Tuning*. Apr 2023. URL: https://arxiv.org/abs/2304.08485

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Björn Ommer. *High - Resolution Image Synthesis with Latent Diffusion Models*. December 2021. URL: https://arxiv.org/abs/2112.10752