Review

Not peer-reviewed version

Multi-Agent Systems and Their Evolution: A Comparative Survey

Asifullah Khan *, Hamna Asif, Maha Tariq, Hania Khan, Inaya Imran, Nayab Ibrahim, Aqsa Asif, Zunaira Rauf, Aleesha Zainab, Saleha Jamshed

Posted Date: 4 June 2026

doi: 10.20944/preprints202606.0358.v1

Keywords: large language models (LLMs); multi-agent systems (MAS); Autogen; Langroid; MetaGPT; generative AI; natural language understanding

 doi:10.20944/preprints202606.0358.v1

Review

Multi-Agent Systems and Their Evolution: A Comparative Survey

Asifullah Khan [1,2,3,*], Hamna Asif [1], Maha Tariq [1], Hania Khan [1], Inaya Imran [1], Nayab Ibrahim [1], Aqsa Asif [1,2], Zunaira Rauf [4], Aleesha Zainab [1,3] and Saleha Jamshed [1]

[1] Pattern Recognition Lab, DCIS, PIEAS, Nilore, Islamabad, 45650, Pakistan.
[2] PIEAS Artificial Intelligence Center (PAIC), PIEAS, Nilore, Islamabad, 45650, Pakistan.
[3] Deep Learning Lab, Center for Mathematical Sciences, PIEAS, Nilore, Islamabad, 45650, Pakistan.
[4] Department of Computer Science Air University Islamabad, Aerospace & Aviation Campus Kamra, Pakistan
* Correspondence: asif@pieas.edu.pk

Abstract

Large language models (LLMs) have revolutionized and have had significant impact on diverse domains such as healthcare, software development and autonomous systems by enabling natural language understanding and reasoning. However, single agent architectures limit their potential due to their non-collaborative nature and the reduced capability to perform complex, multi-disciplinary tasks which require teamwork, role division and adaptive decision making. To counter these shortcomings, Multi-Agent Systems (MAS) have been developed into a platform of contemporary artificial intelligence that allow the autonomous agents to interact, reason and communicate with dynamic and complex environments. Accompanied by the growth of experimentation and implementation of LLMs and Generative AI, MAS frameworks have been applied to real-world applications increasingly. This survey offers an in-depth and comparative study of three innovative and advanced MAS frameworks; Autogen, Langroid and MetaGPT. It delves into their architectural design, communication standards, scalability, applicability and their integration into the rising real world technologies. It presents standard benchmark criteria and performance measures (latency, throughput and memory utilization) through detailed case studies across diverse application domains such as e-commerce, medicine and AI-assisted software engineering. Moreover, it highlights important issues like explainability, security, computational cost and human-in-the-loop requirement in designing such models. Being a synthesis of theoretical developments and practical implementation experiences, it provides a systematic decision-making guide and serves as a basis of further MAS research and development.

Keywords: large language models (LLMs); multi-agent systems (MAS); Autogen; Langroid; MetaGPT; generative AI; natural language understanding

1. Introduction

In Artificial Intelligence (AI), a branch of computer science, intelligent agents have been designed to perceive their environment, make relevant decisions, and act accordingly to achieve some predetermined goals [1]. These agents have demonstrated progressive success not only in robotics but also in decision support systems. However, due to the increasing complexity of real-world problems, the drawbacks of single-agent systems have become very evident. A single agent working alone may often not be able to perform tasks that require distributed reasoning, parallel processing, or coordination among competing objectives [2].

To overcome such challenges, the researchers introduced the concept of Multi-Agent Systems (MAS). These systems are a combination of many autonomous agents that interact and work together to address the problems that cannot be effectively solved by any single agent. MAS has made

 doi:10.20944/preprints202606.0358.v1

significant progress in artificial intelligence and enables scalable, flexible and resilient solutions to a wide range of domains. These systems are able to perform more efficiently and responsively to uncertain environments by integrating multiple and specialized capabilities of individual agents [3]. To simplify their design and development, a number of MAS frameworks have been developed which provide standard infrastructures, program models and middleware support. These frameworks help in the modeling and deployment process as they offer reusable elements to communicate, coordinate and interact with the environment. This accelerates experimentation and aids in bridging the distance between the theoretical studies and application [4].

Over the past few years, the fast-paced evolution in terms of LLMs has re-established AI, particularly NLP [5]. LLMs, such as GPT and its successors, exhibit powerful capabilities in understanding and generating human-like language [6]. Although single models are designed in a traditional way, multi-agent paradigms are currently being integrated with LLMs as they are addressing more complex issues of AI. It is the integration of LLM capabilities with MAS architectures that allows state-of-the-art problem solving [7].

The introduction of LLMs enables MAS to chat, negotiate and cooperate on higher ranks, thus increasing the range of activities that they can manage with very little human intervention. The problem of scalability and efficiency in MAS structures is extremely topical in connection with the further expansion of AI applications in various industries. Thus, this paper explores existing issues, constraints, and technological implementations in MAS models to enhance their development and implementation in various applications [8].

1.1. Evolution of MAS Frameworks

MAS originated from research in DAI and the extended use of agent-based technologies. In time, this area evolved into a significant area within computer science focused on the study of autonomous entities called agents. Early research in the late 1980s and the 1990s was primarily based on rule-based and heuristic approaches, where agents acted within established logic structures and decision trees [9]. Even though these early MAS frameworks provided the basics for agent-based modeling, they suffered from serious problems concerning adaptability and scalability in dynamic, unpredictable environments limiting their effectiveness in dynamic real-world environments [10].

While MAS research was progressing, NLP also matured from early rule-based systems of the 1950s–60s and statistical models of the 1980s–90s to more sophisticated approaches, such as word embeddings in the mid-2000s, thus addressing the challenge of more natural interaction between agent and envoronment. The introduction of RNNLMs in 2010 [11] and Google's GNMT system in 2016 [12] led to significant improvements with respect to modeling sequences. Significant advancements in the field were the introduction of Transformer-based architectures, beginning with models like BERT and GPT-1 [13]. In addition to improving language understanding, these advancements also enabled agents to communicate, coordinate, and collaborate more effectively, allowing MAS to transform from simple task-sharing systems into frameworks capable of complex negotiation, decision-making, and collective problem-solving.

Reinforcement learning with LLM has changed the way industries are being operated, allowing agents to learn, use experience to adjust their strategies and make superior decisions as they progress [14]. With the help of RL, it is possible to train agents in virtual settings with the mechanisms of reward and punishment to optimize their behavior. Recently, advanced LLMs such as GPT-3, GPT-4, GPT-4o, Claude 3, and Gemini 1.5 have further improved the capabilities and communication of MAS in NLP and other complex areas [15,16].

This trend has motivated the creation of new next-generation MAS systems such as AutoGen [17], MetaGPT [18], and Langroid [19] that formally use LLMs as intelligent agents. These types of frameworks permit scalable coordination, effective coordination of tasks, and collaborative processes in which agents can reason, plan, and act jointly; therefore, they are the latest in MAS literature and practice implementation.

 doi:10.20944/preprints202606.0358.v1

Although such developments are made, there are still some important challenges remain. First but not the least, the majority of the state-of-the-art solutions in this regard fail to offer a clear picture of their scalability, coordination and integrations with the modern AI technologies because of the absence of standardized benchmarks, structured comparisons, and universally accepted evaluation metrics [20]. Futhurmore, related surveys do not thoroughly map these gaps along various frameworks [2,7,16]. Considering such a constraint, this survey concentrates on the pros and cons of the newest generation of MAS frameworks such as AutoGen [17], MetaGPT [18], and Langroid [19] in order to learn about their practical implementation.

This paper will present a summary of the current MAS frameworks and especially highlight their design principles and their application in different fields. Section 2 outlines the existing frameworks, their main features, areas of use, and the increasing role of LLMs in the development of MAS. Section 3 outlines the evaluation methodology, and compares these frameworks. Section 4 discusses case studies that are real world applications. Sections 5 and 6 address domain-specific applications and provide a summary of the future, including some of the newer technologies developed, the challenges, and potential opportunities. This article is focused on bridging the gap between the theory of MAS and its practice through the structured and comparative assessment.

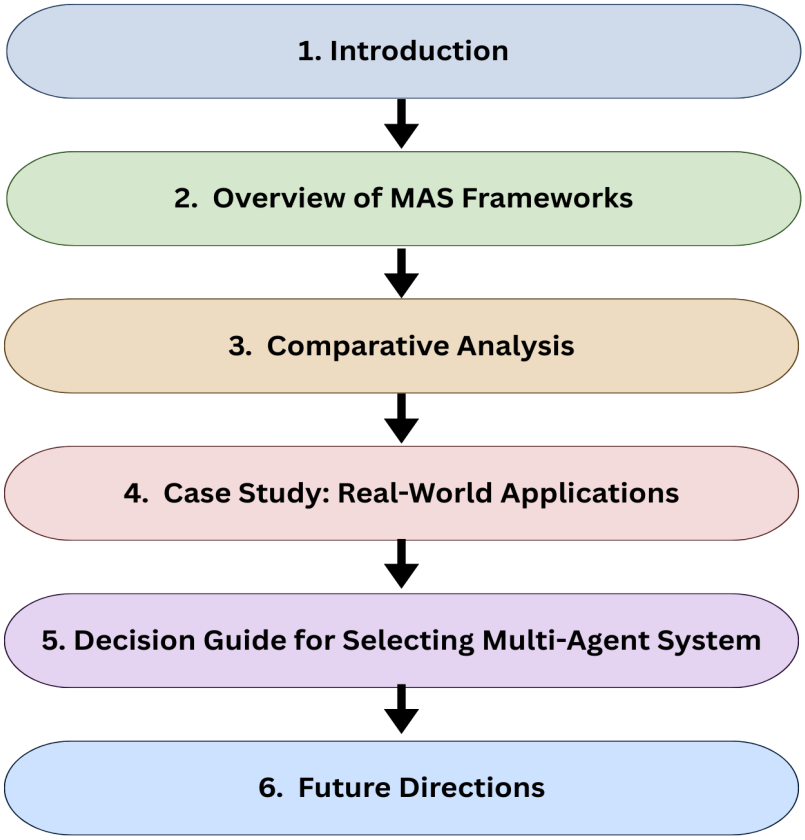


Figure 1. Layout of the different sections of the survey paper.

Table 1. Table of abbreviations.

Abbreviation	Definition
MAS	Multiagent Systems
AI	Artificial Intelligence
RAG	Retrieval-Augmented Generation
IE	Information Extraction
NER	Named Entity Recognition

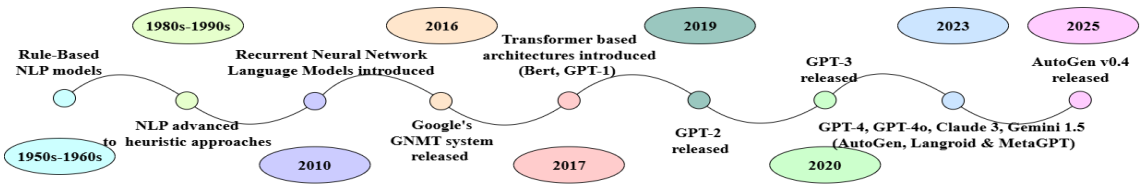
 doi:10.20944/preprints202606.0358.v1

SRL	Semantic Role Labelling
KBP	Knowledge Base Population
SOP	Standard Operating Procedures
MAPF	Multi-agent Pathfinding
IoT	Internet of Things
XAI	Explainable Artificial Intelligence
GDRP	General Data Protection Regulation
CCPA	California Consumer Privacy Act
HPC	High Performance Computing
API	Application Programming Interface
REST	Representational State Transfer
RL	Reinforcement Learning
Deep RL	Deep Reinforcement Learning
RNNLM	Recurrent Neural Network Language Model
API	Application Programming Interface
META	Model Exclusive Task Arithmetic
AI	Artificial Intelligence
LLM	Large Language Model
NLP	Natural Language Processing
MLLM	Multimodal Large Language Model

2. Overview of MAS Frameworks

Multi-Agent System (MAS) architectures allow organizing a systematic environment in which a number of intelligent agents may collaborate autonomously to address complex tasks in software systems [21]. These systems become intelligent in general through the interactions of these agents, each with its own goals, beliefs and local knowledge [10]. The first MAS systems, created during the 1980s and 1990s, were based on rule-based and symbolic methods. Systems like Jason were based on the Belief-Desire-Intention (BDI) model, and JADE on the FIPA standards [22–24]. These models had been tested to work well in stable environments, but lacked scalability and flexibility in real-world dynamic environments [25]. MAS structures have been changing with time and have resorted to the use of Large Language Models (LLMs) and machine learning methods. The current systems like AutoGen, MetaGPT, and Langroid allow the agents to cooperate in natural language, do general planning and work on unstructured problems without depending on strict rule-based systems. This has made MAS to become more dynamic, scalable, and simple to expand.

In 2026, there are other new developments that further underscore the shift from research-oriented prototypes to enterprise-level MAS implementations. New ideas like agentic AI focus on how to coordinate a number of specialized agents in complex workflows. Simultaneously, novel frameworks like MASFactory [113] and MASFly [114] include more sophisticated functionalities such as orchestration through graphs, dynamic adaptation and interaction with humans, enhancing flexibility and reliability. Moreover, there is a growing focus on governance, scalability, and resilience of the system, which indicates the rising demand of reliable and production-ready multi-agent systems [116]. Consequently, contemporary MAS models can coordinate in real-time and can be effectively deployed to solve sophisticated problems like software development, analyzing supply chains, and data-driven decision-making [26].



 doi:10.20944/preprints202606.0358.v1

Figure 2. Evolutionary flowchart illustrating the development of Multi-Agent System (MAS) agents.

2.1. Framework Architectures

System architecture is the backbone of the system; it determines the development process, provides reliability, and future scalability [27,28]. It is used in Multi-Agent Systems (MAS) to describe the organization, interaction and execution of agents. MAS architectures are commonly categorized along with the organizational paradigm, such as centralized and decentralized control. More complex classifications include the aspects of hierarchy, homogeneity of agents and pattern of communication, which offer a systematic analysis framework. Some of the common organizational models are teacher-student, leader-follower, sequential coordination and colony-based systems which provide various ways of collaborating and distributing tasks.

2.1.1. Centralized and Decentralized Architectures

The main aspect of centralized architecture of MAS is that it is a unit of control which collects information worldwide and coordinates agent activities. This eases coordination and provides uniformity but presents constraints like scalability concerns, bottlenecks and single points of failure [29,30]. The early MAS models and components, such as group chat managers, tend to follow this model.

The decentralized designs, on the other hand, spread out the control to the autonomous agents which utilize local information and peer communication [31]. This increases items such as flexibility, scalability and robustness, however, it complicates coordination and it becomes hard to gain global consistency [32]. Moreover, other problems that the decentralized learning systems should address are non-stationarity and communication overhead.

2.1.2. Homogeneous and Heterogeneous Agent Systems

Another type of MAS is by agent similarity. Homogenous systems consist of agents loved by capabilities making them easy to design, coordinate and manage. Their simplicity has made them a major subject of study [33]. Conversely, heterogeneous systems are composed of agents with dissimilar functions as well as specialized capabilities. They are very difficult to design and organize but can be more effective in case of task specific situations [34,35]. The techniques such as asynchronous advantage actor-critic (A3C), which facilitate agent-specific learning and adaptation, support such environments [36].

2.1.3. Hierarchical and Flat Organizational Styles

The framework of MAS organization defines how agents are organized and how the system is coordinated in order to achieve system objectives. Hierarchical architectures (or holarchies) organize the agents into layers, where the higher level agents deal with planning and coordination whereas the lower level deal with implementation. Such a structure simplifies and streamlines scaling and decision making, but can cause rigidity and cascading failures [37]. Below are the flat (peer to peer) architectures in which the agents are equalized, and are encouraged to be independent and strong. However, they are likely to experience problems related to communication overhead and global coherence. This strategy is normally witnessed in swarm systems or colony systems. Sequential (series-based) coordination can also be adopted by both hierarchical and flat structures, in which tasks are handled in a linear, stepwise, assembly-line fashion [37].

2.2. Interaction Mechanism

Interaction mechanisms define the ways agents strategize, exchange information and make decisions in a MAS. They describe the communication between agents, flow of information, and the emergence of decisions at the system level. MAS frameworks can accommodate a variety of

 doi:10.20944/preprints202606.0358.v1

interaction styles, from formal, protocol-based interaction to flexible natural language interaction [38].

2.2.1. Communication Paradigm

Communication paradigms define how agents share information. In direct communication (message passing), agents transmit explicit messages in a predefined protocol like Request, Inform, or Propose [39,40]. It is a transparent approach which is common in decision-making and is achieved by negotiation, voting, or delegation [41]. They may be traditional protocols, such as FIPA ACL and KQML, or recent protocols such as Agent-to-Agent (A2A) protocols and group chat systems of AutoGen [42,43].

In indirect (environment-based) communication, agents communicate by means of a common environment or memory [44,45]. Actions, e.g., updating a shared task list, are signals to other agents. It results in more reactive and emergent behavior. Agents notice and react to environmental changes [46,47]. Common examples include blackboard systems, shared memory models, and MetaGPT's document-based workflow.

2.2.2. Interaction Protocol

Interaction protocols establish the meaning and structure of communication. Standardized protocols employ rigid syntax and semantics, which are based on predefined performatives, to provide transparent and dependable interactions. These techniques are typical in architectures such as JADE and communication standards at the enterprise level, and may be based on formal coordination strategies such as the Contract Net Protocol [24].

On the other hand, the (natural language) protocols of the LLM can be used to communicate using a flexible and human-friendly language. LLMs have the ability to infer the user's intent and contextual information [41]. This allows for AutoGen, CrewAI, and MetaGPT's style of initiation, negotiation, delegation and role transition to thrive.

2.2.3. Orchestration Mode

The way the flow of interactions is regulated is called orchestration. The communication is centralized i.e., the communication is regulated by some coordination which supervises who should get next. This guarantees an organized performance and convenient conflict management, such as the AutoGen GroupChatManager or organizing the workflows of MetaGPT tasks [48]. Tasks In decentralized (peer-to-peer) orchestration agents: there is no interaction with a controller. Any agent can take the initiative or respond to local knowledge, leading to more adaptive and resilient systems. However, it is more intricate, and coordination typically relies on broadcast systems or autonomy to select roles [49,50].

2.3. Knowledge Representation

Knowledge representation is the most essential aspect of MAS that determines the way information is stored, structured and utilized to reason and collaborate [51]. It can be generally categorized as implicit, explicit and contextual knowledge. Stored in the internal representation of an agent's domain knowledge, world knowledge and reasoning abilities is known as implicit knowledge [52]. It helps in problem solving, language understanding and/or decision making but is not visible physically. On the other hand, explicit knowledge is structured and testable; and is represented in the form of knowledge graphs, ontologies, vector databases or rule-based systems, providing them with consistency and reliability [53]. Contextual (episodic) knowledge consists of short term information like the history of messages, task context, intermediate states and provides continuity to interactions.

It also plays a role in the effective utilization of knowledge organization and sharing. The semantic organization utilizes the data to formulate the graphs or taxonomies, which are simpler to

 doi:10.20944/preprints202606.0358.v1

query and understand the relationships. Procedural organization (the output of one worker is the input for another worker) and role-based organization (definitely assigned knowledge and responsibility given to agents) are two typologies of workflows. It also guarantees consistent system uniformity by sharing knowledge which guarantees consistency in the handoffs, relay of messages and shared memory space.

Lastly, such types of knowledge support us in effective decision making and reasoning. The agents integrate internal information with outside resources to justify themselves (augmented reasoning), rely on situations to enhance coherence and structure their actions to serve collective ends, eradicating redundancy and benefiting system functionality net [55].

3. Recent MAS Frameworks

Multi- Agent system (MAS) systems have significantly made the design, deployment and management of agent based systems very convenient. The early MAS were typically specific in problem solving and inter-organizational coordination but large scale deployment was problematic. To cope with this, scholars presented MAS frameworks, which are organized platforms that offer standardised communication, inbuilt tools and agent interaction management mechanisms [4]. Recent research also indicates the increasing significance of such systems in facilitating scalable, coordinated and intelligent agent-based systems, especially in complex application areas, like software engineering and decision support systems [116].

Initially, these models focused on basic communication and separation of tasks as well as offered very little flexibility. Still, the innovation of Large Language Models (LLMs) resulted in the present models that enable more effective collaboration, adaptive behavior as well as the ability to complete intricate tasks. Similarly, the MAS research has grown beyond the simple communication systems to smart and scalable systems which can enable real world applications. The subsequent passages discuss the latest models, representing the same development.

3.1. AutoGen

AutoGen is an open source multi-agent system to build multi-agent apps by generating dialogue and collaboration using LLMs. It enables both agents, humans and tools, to talk to each other to complete complex tasks, therefore it is applicable in applications such as automated reasoning [21], chatbots and virtual assistants [22].

3.1.1. Agent Model

AutoGen is grounded on the concept of conversable agents, which may be used to interact with users, other agents, and external tools to finish tasks. The agents can be defined in natural language or in code and this provides the developers with the flexibility of defining their behaviour and interactions [56]. Its structure maintains a variety of specialist agents, each of which is adapted to a purpose within the system. Examples would be of an Assistant Agent that communicates with users and processes their tasks, the Proxy Agent that is used to provide access to the outside world, the Tool Agent that executes APIs and external tools and the Coordinator Agent that manages distributed coordination. Human-in-the-Loop Agent is also included to offer the possibility of human intervention in the situations when human judgment is required. The latest extensions have added features such as the Safeguard, Writer and Commander agent that bring more capabilities to the system when it comes to safety monitoring and generation/delegation of content. Its design is both role based and modular based to improve reusability and flexibility of AutoGen in a wide range of applications [58].

 doi:10.20944/preprints202606.0358.v1

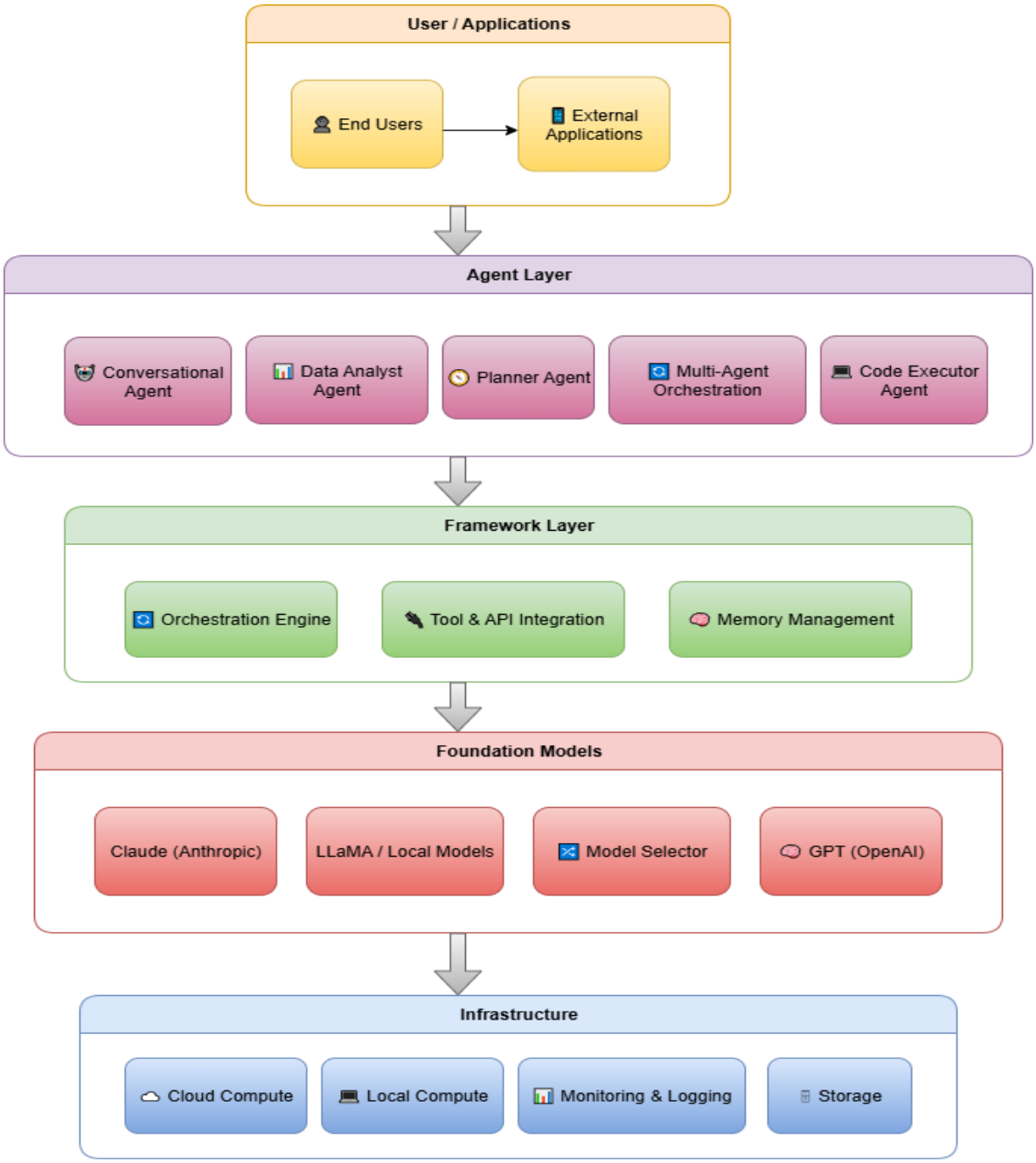


Figure 3. Architectural overview of the AutoGen Framework for multi-agent collaboration.

3.1.2. Communication Protocol

AutoGen is an asynchronous and event driven structured, message based communication model. It enables request response and event-based interactions and allows real-time collaboration.

The agents can take a form of communication through natural language or code and the structure can take a variety of patterns of coordination including sequential, parallel workflow and hierarchical workflow. Interlanguage compatibility (e.g., Python and .The NET) also promotes interoperability [56,59,60].

3.1.3. Knowledge Representation

AutoGen architecture allows easy integration into external tools and APIs (Application Programming Interfaces), allowing agents the access to a multitude of potential knowledge sources, expanding their knowledge bases. With this integration, agents have access to real-time data and are

 doi:10.20944/preprints202606.0358.v1

able to perform calculations and communicate with the outside world, all of which significantly improves the process of problem-solving in many different areas. Queries to databases, calls to specialized computational engines, and access to up-to-date information via web services can effectively enable agents to augment their reasoning capabilities with dynamic, domain-specific knowledge [61].

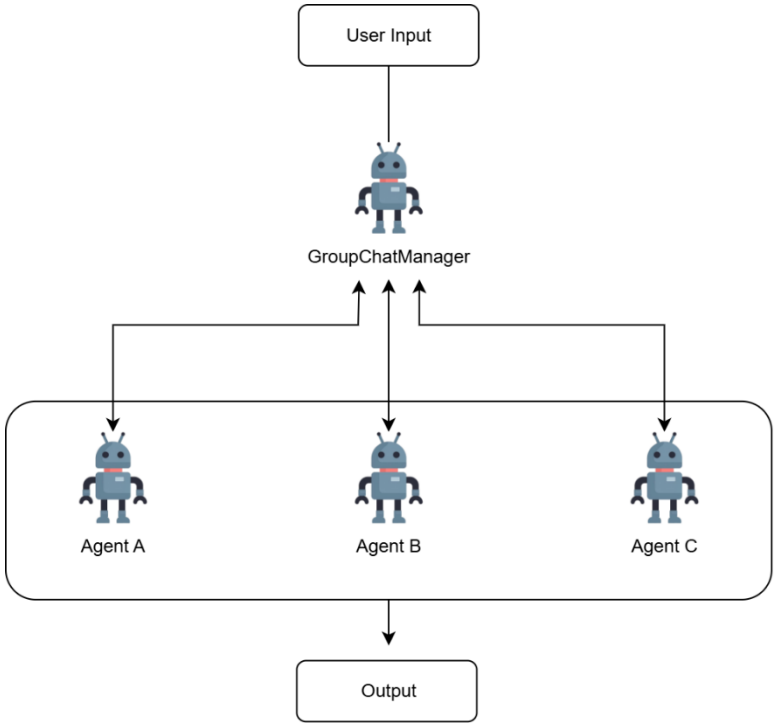


Figure 4. Internal Communication Architecture of AutoGen GroupChat.

3.2. Langroid Framework

Langroid is an open-source system that was designed by the scientists of the University of Wisconsin-Madison and Carnegie Mellon University. It places importance on modularity, extensibility and organized multi-agent cooperation [62].

3.2.1. Agent Model

Langroid is framed on an agent architecture that is modular and extensible, where agents interactively solve tasks in a structured way. The fundamental element is the ChatAgent that combines a Large Language Model (LLM), external tools and vector stores to carry out task-specific actions. ChatAgents work together in contrast to isolated agents, where messages are shared and problems are solved collaboratively. Another prominent characteristic of Langroid is the DocChatAgent, which is a specialized agent that is used in document-based tasks based on Retrieval-Augmented Generation (RAG). It is capable of retrieving the appropriate information in external documents as well as producing context sensitive responses, thus making it suitable for domain-specific applications.

Langroid considers agents to be first-class; that is, each agent has its own conversational state, processes incoming messages, and generates responses as an independent entity. Agents act as message transformers, taking in input, performing reasoning or tool-based operations, and passing the output to other agents. The design gives flexibility and consistency of the multi-agent workflows. In order to address the complex interactions, a tasks-based coordination model and Task class is proposed by Langroid. Agents are assigned with particular roles, goals or instructions in a task. The framework helps to decompose a problem into smaller sub tasks, which the agents can handle, and then recursively repeats this process. These interactions get facilitated by using the Task.run()

 doi:10.20944/preprints202606.0358.v1

function which provides an organized execution with flexibility. In practice, this modular agents-state-management-hierarchical coordination of tasks combination allows Langroid to scale up, as well as other knowledge-intensive workflows [63,64] to which it was applied.

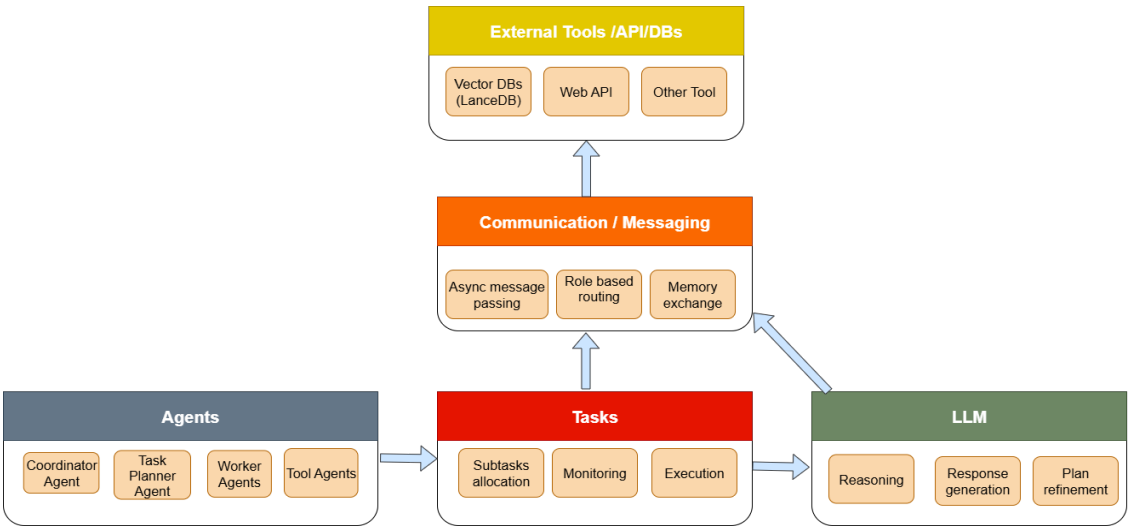


Figure 5. Architectural Overview of Langroid framework.

3.2.2. Communication Protocol

Langroid adopts an asynchronous approach, agents are only able to communicate with each other by exchanging structured messages. Each agent is a message transformer, i.e., it takes in processes with its internal logic (LLM reasoning, tools, or memory) and gives out an output which can be sent to other agents. The communication is managed via responder methods, that is, how the agents respond to various kinds of messages. Such techniques enable the agents to communicate with other agents as well as with external tools, vector stores, and human inputs. This is an organised but adaptable system, which makes sure that communication is formalized yet not limiting to rigid behaviour.

An important element of the communication system created by Langroid is the Task class which controls the flow of interaction. It keeps communication loops under control, and agents adhere to a specific sequence, yet making it possible to respond flexibly. In this way, Langroid facilitates multi-step reasoning, refinement, and problem-solving.

 doi:10.20944/preprints202606.0358.v1

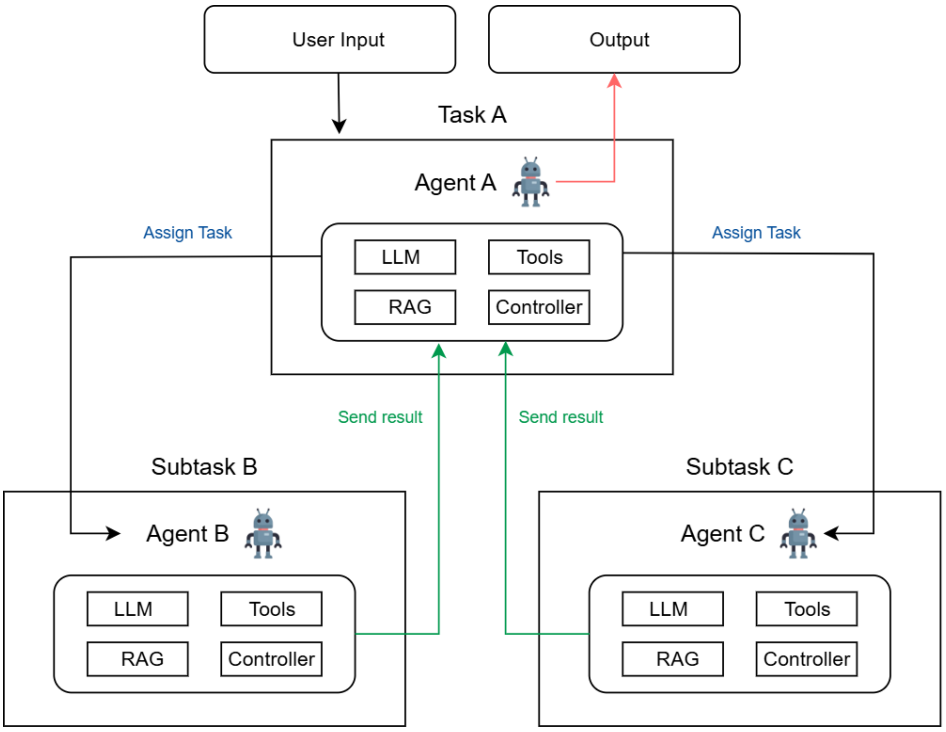


Figure 6. Langroid communication model showing task-driven interactions among tool-augmented agents.

3.2.3. Knowledge Representation

Knowledge representation in Langroid is memory-based in which the agents are able to store, recall and recycle information during interactions. Each agent can have a long-term view of its context, which helps to provide continuity and enable them to act on their decisions.

Integration of the vector databases, such as Qdrant, Chroma, LanceDB, Pinecone and PostgresDB is one of the key features of such framework. They are long term memories, which will allow an agent to remember something that is related in some semantic way, but may not be in the same exact context in which it was learned. This mechanism make sure that the responses obtained are based on the already stored information and not produced purely out of the parameters of the LLM itself.

The other important element of the framework is knowledge sharing. Shared memory spaces allow agents to share information or exchange messages, and in this way, allow collective intelligence. The agents dynamically adapts quickly as tasks change, and thus, they gain better performance of the system.

The integration of the tenacious storage, awareness of context, and collaborative sharing of knowledge offers a solid platform of knowledge management in Langroid to deal with knowledge-intensive and real world applications [19].

3.3. MetaGPT Framework

MetaGPT is an LLM-based multi-agent system that is directly aimed at automated software development. Compared to general-purpose MAS frameworks, MetaGPT uses specialized agents, like product manager, architect, engineer, and QA tester, to mimic structured software development processes [18]. Encoding Standard Operating Procedures (SOPs) into prompt sequences not only allows role-based cooperation and complex task execution but also uses the principles of meta-programming to organize the interaction among agents.

3.3.1. Agent Model

 doi:10.20944/preprints202606.0358.v1

MetaGPT uses a role-based agent model whereby each agent is designated a particular role in the software development lifecycle. The Product Manager collects user requirements, project objectives, and coordinates activities with the goals of business. The Architect decides the overall system structure and choice of suitable technologies to address the functional and technical requirements. The Project Manager will deal with scheduling, tracking of resources and milestones to ensure timely progress. The Engineer transforms requirements into executable code, adding features and correcting bugs and QA Engineer ensures that the achieved output is the right product and accomplishes this by performing tests and ensuring the quality of the final product requirement [65].

This separation of tasks creates a factory-like production process, where the agents are assigned to parallel aspects of a task in an assembly-line paradigm. The agents are able to specialize in areas where they are specialized by dividing complex tasks into small tasks that are specific to various positions that increase efficiency, accuracy and scalability [66]. It is also dynamic and can be changed as the architecture can accommodate the changes in requirement or feedback without trying to halt the whole process. Such an organization of roles and the implementation of particular tasks, makes it possible to guarantee that the framework was capable of studying large and complicated software projects.

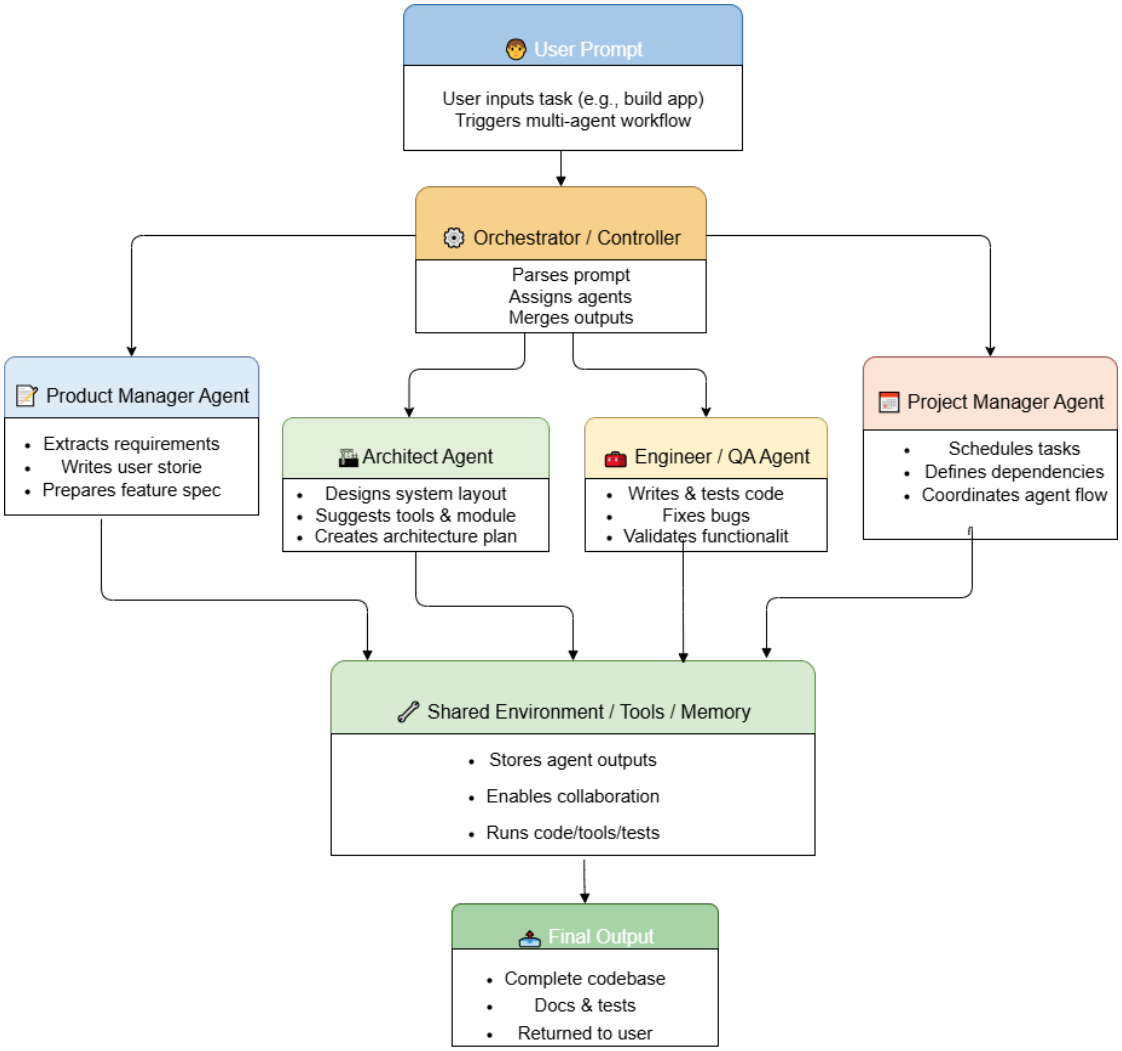


Figure 7. Architectural Overview of MetaGPT Framework.

3.3.2. Communication Protocols

 doi:10.20944/preprints202606.0358.v1

MetaGPT, which is structured by SOP-based protocols, ensures that the flow of communication is tightly defined, both in terms of the order and content of the communication, to ensure that the communication has meaning and maintains consistency. Each agent is running stimuli (prompts) that govern the way to carry out the tasks, maintain order, and minimize error. The sequential dependencies are explicitly coded meaning that agents will check the results of the previous tasks before moving on with their own tasks. Such a mechanism does not only provide accuracy in the execution but also reduces redundancy and promotes quality production. The communication model of the framework is similar to an assembly line, in which the agents plan their communication in a systematic flow, with specialization and joint efficiency. MetaGPT provides a predictable and reliable communication system by merging structured messaging with built-in verification loops that are especially effective in software development processes that demand accurate coordination and quality control [18].

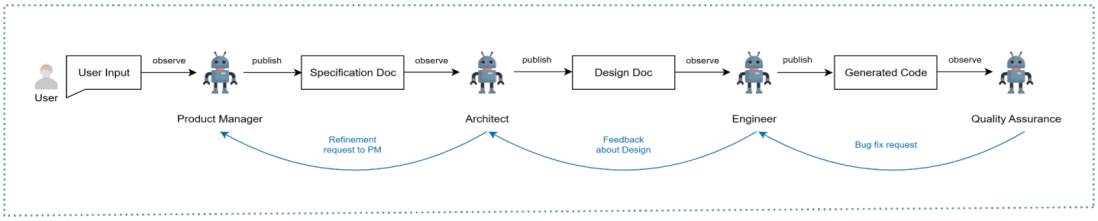


Figure 8. Communication model of MetaGPT showing sequential SOP execution with integrated feedback loops.

3.3.3. Knowledge Representation

In MetaGPT the knowledge is structured based on the role of agents meaning that each agent can only access the information relevant to the duties of the agent. Agents maintain domain knowledge and SOPs in forms of retrievable prompts, which are used to reason and perform tasks. To ensure consistency in context and minimize errors that may occur in the processing of LLM, the framework uses collaborative validation, where the agents check and verify the output of the other agents before being incorporated. Continuous feedback loops enable agents to dynamically revise their knowledge according to new information or corrections and enhance adaptability and accuracy. The framework also capitalizes on the high-level concepts like the local linearity of LLMs to enable easy transitions between task representations and task vector orthogonality to ensure that any role is independent of the other to minimize conflicts and enhance efficiency. Furthermore, learning multiple tasks and model fusion in a secure manner with data independence while retaining sensitive data without compromising the performance is possible in the Model Exclusive Task Arithmetic (META) architecture [67]. The hierarchical knowledge sharing ensures consistency and flexibility of the agents due to the sequential up-to-date information transfer; each agent has his own specialization to play his role and allows the interdependency of the different processes [68].

4. Comparative Analysis

In order to distinguish the use cases and practical merits of AutoGen, MetaGPT and Langroid in multi-agent systems (MAS) in the real world, it is crucial to perform a thorough comparative study of these three concepts systematically. The area which this section will evaluate the frameworks in will encompass various categories including ease of use, performance, customization, communication and scalability. We place the analysis around these criteria to provide a clear point of reference for analysing their design trade offs and their suitability in alternative applications. This comparative view not only makes the researcher and practitioners aware of the difference in architecture between them, but it also will help them to select the best architecture that fits their needs in some applications.

 doi:10.20944/preprints202606.0358.v1

4.1. Ease of Use and Real-World Applicability

Multi-Agent Framework is also important for ease of use, in that ease of design, deployment and adaptation of Agent systems for real-world applications is directly tied. The AutoGen framework is the simplest to work with hence it is the easiest to use amongst the three frameworks. It also offers built-in agent creation and interaction management capabilities that can reduce a great deal of implementation effort, and allows users to use more powerful problem solving, rather than more-trivial system design.

Langroid, however, not only places a high value on control and flexibility but on more complexity as well. Where users want a finer-grained control over agent actions and system customisation, it is more technically demanding to establish agents and administration of communication. In turn, MetaGPT is an application-oriented product. In structured workflow environment such as software development or research work, it allows pre-defined agent role to facilitate the easy set up of the system, and facilitates fast deployment. However, it is not as general-purpose as more common systems, like Langroid.

From a practical point of view, for projects with a relatively medium-sized agent system, high automation of tasks, ability to seamlessly integrate with REST APIs and external tools, and integration with other language models that can be used in the project, AutoGen can be useful. It also has an event-based communication model that helps carry out the asynchronous execution. Langroid is written to operate lightweight and in a stable manner meaning it can be used in resource-constrained environments, such as Internet of Things (IoT) devices and mobile platforms. It can be used to address natural language tasks in a low-cost manner, by combining it with large language models and retrieval-augmented generation (RAG) pipelines. In contrast, MetaGPT is geared toward complex reasoning and decision-making tasks, like medical analysis, monetary organizing, and study workflows. These typically require a graphics card or cloud resources and work better when there is a greater need.

Flexibility and adaptiveness is also characteristic of the frameworks. Unlike other tools, AutoGen supports multiple interaction modes and is easy to integrate language models, human input, external tools and even human-in-the-loop to provide oversight of other models when necessary. The module-based extensibility and modularity of Langroid make it easier and more versatile to integrate it into existing NLP environments such as SpaCy and Hugging Face. The differences in the performance and efficiency of these frameworks are further illustrated in the main benchmarks in Table 2. Its ability to work with RAG pipelines renders it especially useful to data-intensive and multilingual applications. While less flexible, MetaGPT is very well structured in terms of collaboration, allowing it to work in highly-structured workflows with standardized outputs and explicit task division.

Table 2. Quantitative Benchmark Comparison of Multi-Agent LLM Frameworks.

Aspect	MetaGPT	AutoGen	Langroid
Benchmark Tasks	HumanEval, MBPP, SoftwareDev	MATH (level-5), ALFWorld, OptiGuide, NaturalQuestions	Not directly reported
Pass@1 (HumanEval)	85.9%	Not directly reported	Not directly reported
Pass@1 (MBPP)	87.7%	Not directly reported	Not directly reported
Math Task Success (MATH Level-5)	Not reported	69.48% (vs 52.5% ChatGPT+Code Interpreter)	Not directly reported
ALFWorld Task Success	Not evaluated	77% (best of 3) with 3 agents vs 54% (baseline)	Not directly reported
Executability Score	3.75 / 4 (SoftwareDev tasks)	Not directly reported	Not directly reported

 doi:10.20944/preprints202606.0358.v1

Execution Time (SoftwareDev)	503 sec	Not available	Not directly reported
Token Usage (per code line)	124.3	Not directly reported	Not directly reported
Human Revision Cost	0.83 corrections (avg)	3–5× fewer interactions than ChatGPT+Code Interpreter	Not directly reported
Multi-agent Coding F1 (unsafe code)	Not reported	+8% (GPT-4) and +35% (GPT-3.5) over single-agent	Not directly reported
Dynamic Group Chat	Not supported explicitly	Built-in support	Supported (via Agent classes + Message handling)
Tool Use (e.g., code exec)	Engineer agent executes code	Tool-backed agents support code and function execution	Optional tool use; user-defined actions or external API calls
Structured Communication	SOPs + shared message pool	Supports both structured and unstructured interactions	Uses message classes and task loops for communication
Custom Agent Design	Fixed roles (PM, Architect, etc.)	Fully customizable via code/natural language	Highly modular; agents defined via Python classes
Feedback Mechanism	Executable feedback loop improves results	Safeguard & interactive feedback via agents	Not explicitly defined; users can script response loops
Application Examples	Software development (end-to-end project dev)	Math, Q&A, Retrieval-Augmented Code, Chess, Decision-Making	Summarization, tool-augmented agents, embedding search

4.2. Performance Metrics

By comparing the three multi-agent frameworks for the next generation with the most relevant evaluation metrics, we draw some conclusions about their respective advantages and disadvantages. These were evaluated using 4 metrics: latency, throughput, memory usage and scalability, all of which capture aspects of performance and resource utilisation. Each metric shows that a framework is likely to be able to support a complex workflow composed of multiple agents and real-world deployment scenarios.

4.2.1. Latency

Latency is defined as the delay between when an agent receives an input or task and when it generates a response. Simply put, it is a measure of responsiveness of an agent in performing tasks [69]. AutoGen exhibits a higher latency with a larger number of agents, primarily because of communication overhead (2 agents = 5-8 seconds, 4 agents = 22-25 seconds). MetaGPT lacks publicly available benchmark results, but it has an advantage of its role-based execution model that can lower the response time by splitting subtasks, although the performance remains dependent on the complexity of the task. Langroid also lacks formal latency reports, but the caching techniques (e.g., Redis, Momento) and RAG pipelines indicate that it can be more effective in text-intensive work. The performance is highly dependent on the particular environment [70].

4.2.2. Throughput

 doi:10.20944/preprints202606.0358.v1

In MAS, throughput refers to the rate at which messages are processed or agents finish their jobs in a given time frame. It is a measure of the overall capacity and efficiency of the system to work. Increased throughput will mean enhanced performance and speed of tasks within the agents. The throughput of AutoGen decreases with the number of agents because the coordination overhead can cause an execution bottleneck. The throughput of MetaGPT can be influenced by the task decomposition and inter-agent coordination complexity, which can impact the performance in large-scale applications. Although it does not have quantitative data, Langroid is expected to work well in workflows that require a lot of retrieval due to its compatibility with other tools such as LangChain, but its efficiency is dependent on the task and the available resources [71].

4.2.3. Memory Usage

The memory usage in a MAS platform is an indication of the size of the underlying language models and the complexity of interaction amongst the agents. High memory demands limit scalability and hinder deployment in environments with limited resources.

AutoGen uses caching and error-handling agents that are based on the use of LLM. This may create moderate- high memory consumption with an increase in the number of agents. MetaGPT minimizes the number of unnecessary computations by encoding workflows into prompt sequences and SOPs, yet its hierarchical architecture makes the memory usage grow exponentially with the complexity of tasks [7]. To reduce reliance on internal memory, Langroid employs RAG pipelines and caching, like Redis and Momento, to enhance efficiency, but quantitative benchmarks are not available [72].

4.2.4. Scalability

Scalability demonstrates the ability of the framework to deal with the number of increasing agents and the complexity of tasks. It informs whether a system can scale out of small-scale prototypes to large and production-level deployments. Scalable frameworks are more generally applicable in fields, whereas poorly scalable frameworks can only be used in high-demand contexts. AutoGen demonstrates a high scalability to the workload in the scenario of dynamically changing multi-agent tasks but the latency also increases with the number of agents and makes it better adapted to moderate than strict real-time systems. MetaGPT is a scalable system that is based on role specialization and organised cooperation, which is effective in well-structured workflows, including software development. Nevertheless, when dealing with large projects, that is, a great number of agents, the computational overhead may take place. Langroid can be used with multi-agent configurations and is practical to NLP tasks based on retrieval, but its behavior with high-concurrent conditions has not been evaluated methodically so far.

Altogether, latency and throughput show the trade-offs between responsiveness and task-handling capacity, memory usage and scalability demonstrate the efficiency and scalability of each framework. AutoGen is moderately-scaled, Langroid is lightweight, and MetaGPT is moderately scaled and supports structured reasoning at the cost of higher computational demands. Combined, these measures give one a comprehensive perspective of the advantages and weaknesses of next-generation MAS frameworks [73].

4.3. Customization Capabilities

One of the essential considerations when analyzing multi-agent frameworks is customization, which defines the extent to which systems can be customized to tackle different tasks and workflows. AutoGen is highly flexible with its Conversable Agents, which lets developers change the behavior of its agents, add new external APIs, and create their own decision-making approaches. Its hierarchical and parallel conversation flows support allow it to effectively manage group interactions and nested subtasks without affecting the underlying system logic [74]. Also, AutoGen Studio is a low-code environment that allows users to design and execute workflows, with drag-and-drop user

 doi:10.20944/preprints202606.0358.v1

interfaces and real-time debugging tools, dramatically reducing the barrier for non-technical users. Its effectiveness is also supported by empirical studies which show a decrease in code complexity in scenarios where there are multiple agents in the automation [57].

MetaGPT, however, adheres to a more structured and systematic approach to customization, emphasizing the importance of roles. Agents are configured to work based on established standard operating procedures (SOPs), with users assigning roles like product manager, engineer or reviewer, in reflection of real life processes. The framework also calls for "feedback loops" which will allow the agents to test and adjust their outputs to help make them more reliable and consistent. In its publish subscribe way of communicating it guarantees agents only listen to relevant information, thus improving the efficiency of coordination. The formalized customization makes MetaGPT particularly versatile in a group context, in which roles can be specialized and feedback is crucial [75].

Again, customization, which is based on modules and communication, is emphasized by Langroid. The agents perform sequential transformers in a structured messaging passing system, where an agent will process information successively and refine it. It has a narrow task distribution system to ensure that the information is transferred to appropriate agents, optimizing the processing and coordination efficiency while avoiding unnecessary overhead in communication. Furthermore, Langroid makes a retrieval augmented generation (RAG) part of their working process and brings into their text the "real" information and enriches it with greater factual accuracy. It can open up access to create the knowledge dynamically, and is incorporated completely with APIs, databases and so on things, that means it can create the actual and context-aware agent interface with the knowledge.

4.4. Communication Mechanisms

Communication mechanisms play crucial roles in systems that involve multiple agents to coordinate and get tasks done; the efficacy with which they do this; and their effectiveness. The structured and conversation driven architecture of AutoGen relies on a structured and predefined web of interactions between the agents that guides their behavior, rather than an open conversation. The main part of this design could be GroupChatManager, which controls the turn taking between the agents, the routing of messages between agents, and the shared context between the agents. Such coordinated co-operation enables AutoGen to deal with complex, multi-step arguments, as well as consistency during dealings [56]. In addition, external tools are also integrated in the communication channel itself and can be executed by the agent in the middle of the interaction to call functions. This feature will improve the trustworthiness and reduce the rate of error when performing tasks [69].

The communication between the operators of the project happens in a communicate based on the "hierarchy" which is based on the project's Step-by-Step procedure (Standard Operating Procedure, SOP). It is also implemented with a publish-subscribe messaging system, in which each agent will be notified only of relevant information that is just pertinent to its task based on its role assigned, thus minimizing redundancy and consequently context drift when collaborating with other agents [18]. Furthermore, MetaGPT has an iteration-verification system, which means that task-specific agents will continually adjust and optimize the output before presentation to the end user. The same quality control for the workflow, and increased credibility of the results, with this hierarchical validation approach.

Instead, Langroid uses a message-passing approach which uses a hierarchy that treats the agents as sequential transformers in which the information at each step is enriched. Its mechanism of task delegation makes sure that communication is limited to those agents who are directly involved in a particular subtask, enhancing efficiency of coordination and minimizing unnecessary overheads [19]. Langroid also facilitates better communication using retrieval-augmented generation (RAG) such that the agents answer using the knowledge sources. This, along with real-time API and database integration, allows agents to utilize real-time contextually informative data, providing that framework particular worth in the applications where the informational nature is vital.

 doi:10.20944/preprints202606.0358.v1

4.5. Agent Programming Paradigms

Agent programming paradigms lay the foundation for the design of agents, the communication between agents and how the tasks are run in a multi-agent system. There are other recent advances such as AutoGen, MetaGPT and Langroid, which point towards other but related paths in this research area. This is a hybrid of the reactive and deliberative programming models used by agents that can react to environmental inputs dynamically, and plan in a structured, goal-directed way. One such trait is it's conversation-based computation model: auto agents communicate with each other rather than the more typical call to function, and dialogue serves as the primary means of coordinating and executing [69].

In contrast to this, MetaGPT follows a role-based approach with greatly defined roles, such as Product Manager, Architect, Engineer and QA. This approach is useful for structuring tasks, decomposing them step by step, performing them as coordinated chains and assessing the results by applying systematic output and continuous improvement while observing the tasks in a structured manner by SOPs. Furthermore, MetaGPT provides the ability to plug in an abundance of models associated with task vectors and does not require a lot of fine-tuning while retaining performance [5].

Langroid, however, is an application that emphasizes modular collaboration based on message. Each agent consists of a language model, external tools and a vector store, and has their very own conversation state. Architecture facilitates a hierarchy of delegation of tasks, with agents breaking down the composition of intricate tasks into smaller, more manageable subtasks, which can again be broken down and delegated. Communication is based on actors and it is loose coupling and scalable coordination among actors [52].

For all these paradigms, trade-offs in employing the design are highlighted: The structured and role-driven execution of the design is emphasized in the paradigm of AutoGen, vs. modular and scalable design of collaboration, emphasized in the paradigm of Langroid, and the paradigm of MetaGPT emphasizes the flexible and adaptive interaction of the design.

4.6. Limitations

Besides performance and functionality, there's the infrastructure requirement and implementation costs of multi-agent framework. A summary of the comparative is provided below. Although they are potentially useful, each one has its own strength and weakness to preclude its abundant use. AutoGen has the same bias issues as its underlying foundation models, which poses questions about fairness and neutrality in domains such as healthcare, hiring, and legal decision-making. Its sealed decision-making process also limits interpretability, as the multi-agent cooperation hinders the tracking of output production. Another main issue is of security vulnerabilities, AutoGen agents have the capability of executing external code and are subject to prompt injection attacks, so it needs security measures such as sandboxing and rigid validation. In addition, its multi-agent structure introduces computational costs, and it is hard to deploy at large scale, in real-time environments, and it requires human supervision since results can be unclear or non-optimal.

In spite of the fact that it is methodical in its role-based architecture, MetaGPT has some limitations. It excludes the chances of cumulative learning in tasks by conducting projects in isolation. It is based on agreed-upon SOPs, which makes it consistent. But it constrains flexibility in unforeseeable, or dynamically changing, circumstances. Combinations of many professional agents also need lots of computing capabilities, and their large-scale deployment is expensive. Moreover, its communication protocols are relatively complicated and thus, it is harder to maintain the high level of information exchange and efficiency loss in the process can negatively affect the overall performance of the system.

The modular and lightweight architecture of Langroid has its own difficulties. Similar to any other system based on LLMs, there is a constraint on the context length of the input, which limits the quantity of information that can be handled during a single interaction. Its outputs are also prone to being brittle since even minor changes in the phrasing of input can yield uneven results, requiring proper handling of input and validation. In turn, latency and token cost are some pragmatic concerns:

 doi:10.20944/preprints202606.0358.v1

the longer and more complicated the interaction, the more likely it is to take up more time (and, subsequently, more cost) will have to be factor to developers when designing. Although Langroid is more efficient in using computational resources than AutoGen and MetaGPT, the heavy dependence on external LLM APIs and tools makes the technology's accessibility limits its scalability both in terms of costs and efficiency.

Table 3. Cost and Infrastructure Comparison of Multi-Agent LLM Frameworks.

Framework	Framework License	LLM/API Cost	Per-run Example Cost	Infrastructure Compute
AutoGen	Open-source (MIT)	Pay per API call (e.g., GPT-4)	~$1.50/session (4 agents, 20 turns)	Local runtime for testing; supports scalable, cloud-based deployment.
MetaGPT	Open-source (MIT)	Pay per API call	$0.20 (simple task)–$2 (full project)	Moderate compute for multi-agent tasks
Langroid	Open-source (MIT)	Pay per API + tools/vector store usage	Not documented yet	Lightweight framework; compute increases with tool use.

5. Case Study: Real-Time Applications of MAS Frameworks

In the real world, MAS has proven to be a valuable paradigm to solve complex, distributed problems in a variety of domains. By enabling multiple intelligent agents to communicate, cooperate, specialize and collaborate, they address several challenges such as enhancing efficiency of systems and enhancing decisions making and its flexibility. This section investigates the various principal domains in which MAS is used, highlighting the improvement of performance, scalability and problem solving in dynamic environments made possible by the collaboration of agents in these environments.

5.1. Industrial Automation and Smart Operations

There is a growing use of MAS within industrial automation to deal with the complex and dynamic workflows of logistics, inventory control and resource allocation. These systems are able to distribute the tasks among specialized agents, so that they coordinate real-time and make good decisions in the large-scale operational environment. Li et al. (2024) examined how AutoGen is applicable in industrial E-commerce to automate warehouses. Their solution used several agents with different functions, such as monitoring the inventory, restocking coordination and workforce planning. This role-based interaction that occurred as a collaboration increased the responsiveness of the system and operational efficiency since it enabled continuous interaction of agents.

Some of the challenges that were successfully managed by the system were real-time inventory management, labor management and robot coordination. The results of the experiments were high workflow efficiency and reduced operating delay. However, several weaknesses were identified such as the synchronization issues in the distributed agents, issues of high workload, scalability limitations, fault tolerance challenges and system integration issues. These results point to the possibilities of MAS to change the way the industrial activities are performed and underline the importance of strong coordination systems and safe integration with enterprise systems.

 doi:10.20944/preprints202606.0358.v1

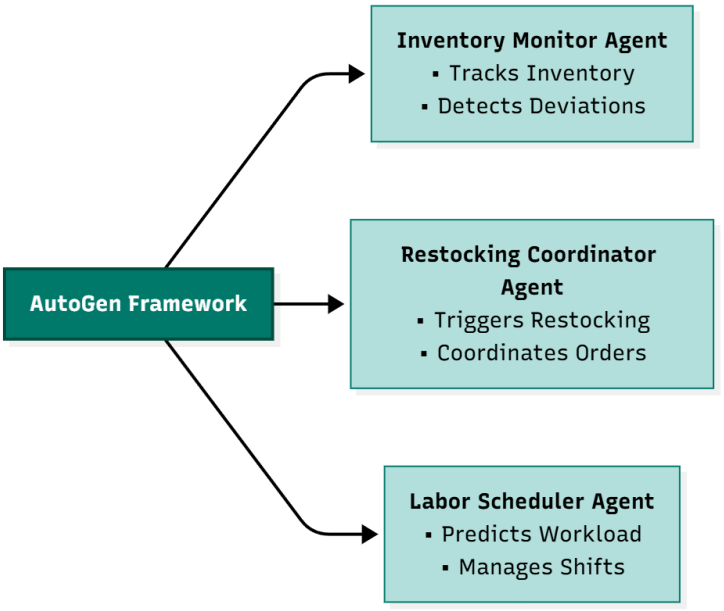


Figure 9. AutoGen Multi-Agent Workflow for Retail Operations.

5.2. Software Engineering and Project Management

MAS have boosted software engineering through the advancement of automation, collaboration, and scalability throughout the entire software development lifecycle [76,78]. These systems allow the autonomous division of tasks, collaborative verification, and dynamic distribution of roles, which leads to greater reliability in large-scale and complex projects [79].

Numerous software development frameworks based on MAS are constructed upon MetaGPT, an AI-inspired model of engineering teams to organize activities like planning a project, writing a program, and testing quality assurance [68,80,81]. Translating Standard Operating Procedures (SOPs) into executable processes, MetaGPT programs organize workflows through specialized roles, such as product manager, software engineer, quality assurance engineer, and system architect [82].

Experimental research also shows that MetaGPT generates better, well-structured, and executable code that requires fewer human edits than the previous systems like ChatDev. This is mainly because it has iterative feedback mechanisms that ensure that the quality of the code and the consistency of the overall development process is boosted [80,83].

Although these benefits exist, there are several drawbacks. The system encounters difficulties in the ability to keep up with the ever-changing and dynamic technologies, changing project requirements, and multi-language development environments [84].

 doi:10.20944/preprints202606.0358.v1

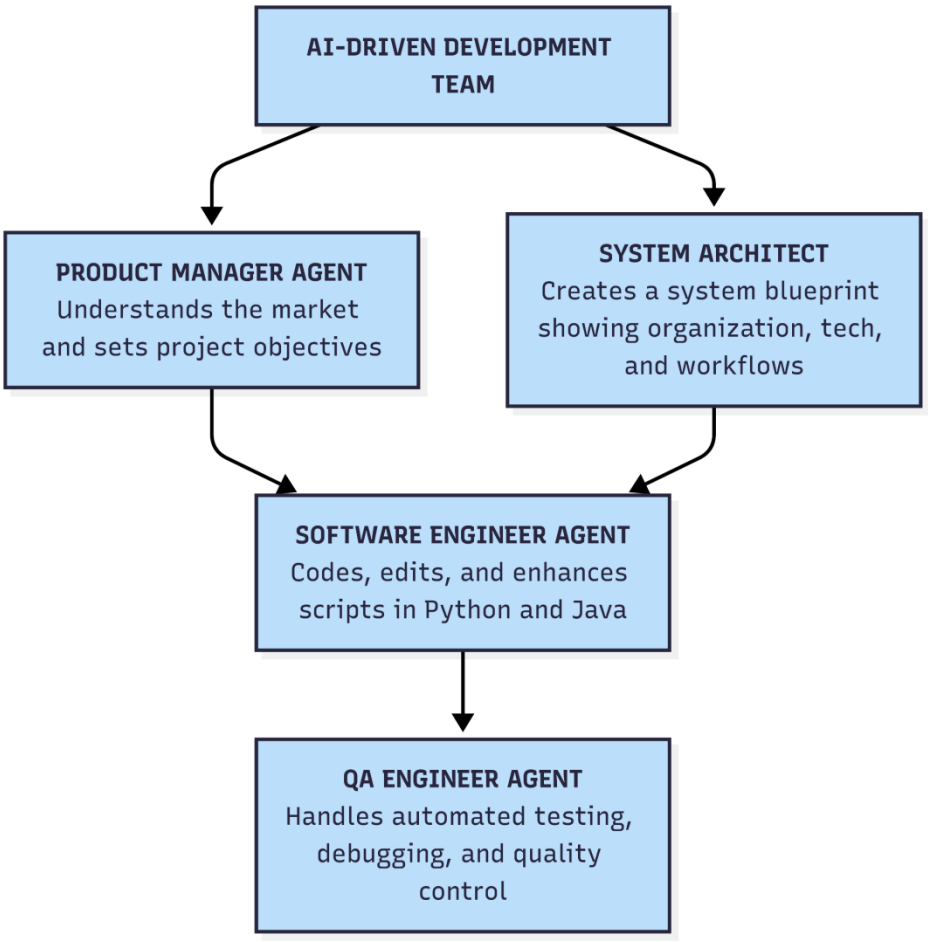


Figure 10. AI-Development team workflow.

5.3. Healthcare and Biomedical Applications

MAS has become increasingly significant and relevant in the field of biomedicine and healthcare, through facilitating intelligent and collaborative decision-making [85]. They have been recently used in disease diagnostics, medical image processing, treatment planning, and patient monitoring [86]. The MAS improves diagnostic accuracy, clinic workflow efficiency and provides responsiveness in real-time by coordinating specialized agents for diagnosis, analysis, and monitoring [87]. Combined with deep learning and IoT technologies, MAS also facilitates adaptive and data-intensive healthcare to offer personalized and ongoing patient attention.

One notable example is MALADE (Multi-agent LLM Architecture in Drug Event Extraction) deployed on the Langroid platform by Choi et al. (2024) [77]. MALADE is the solution to the weaknesses of the conventional adverse drug event (ADE) extraction systems that are commonly time-consuming, labor-intensive, and subject to human error. Task decomposition and medical reliability are lacking in the case of single-model LLMs [87,88]. The system uses three integrated agent-critic pairs to systematically determine the drugs, gather supporting evidence and synthesize the risks involved.

Trained on OMOP (Observational Medical Outcomes Partnership) Ground Truth, MALADE can reach an AUC of 0.85 with GPT-4 Turbo, and 0.90 with GPT-4o, surpassing the current baseline models. Although these are positive outcomes, coordination of agents, managing dependencies, and integration with external tools still poses a challenge, limiting the implementation of MAS on a large scale in pharmacovigilance systems.

 doi:10.20944/preprints202606.0358.v1

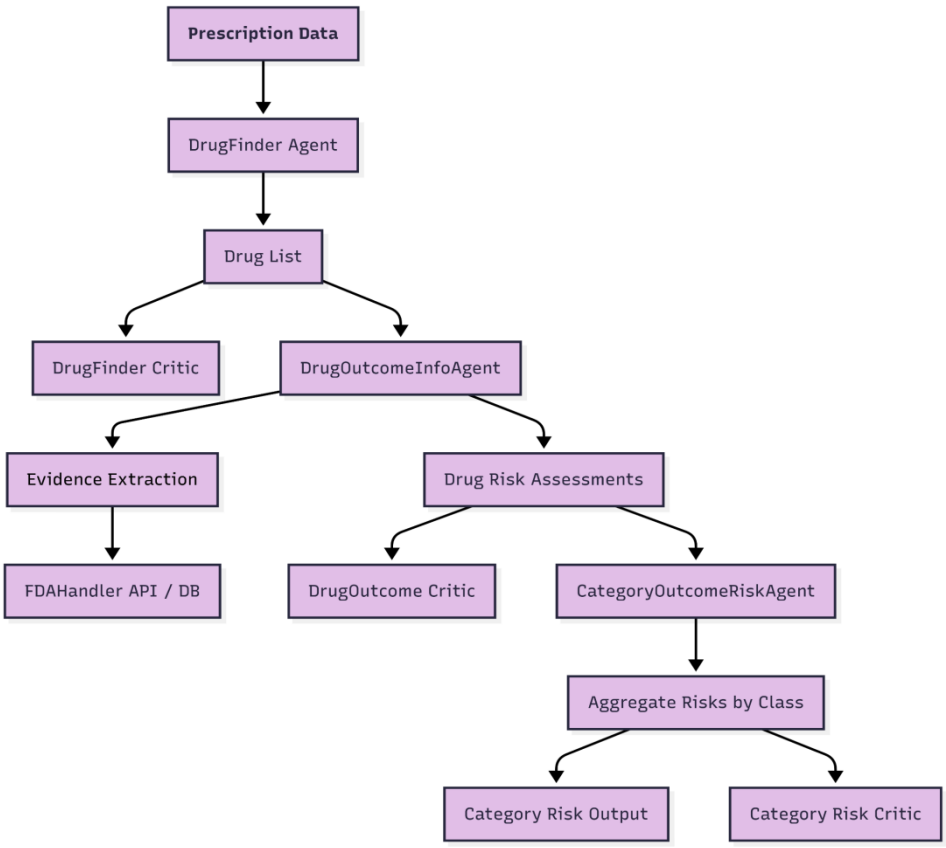


Figure 11. MALADE implementation.

5.4. Finance and Business Intelligence

Multi-agent Systems (MAS) have been widely used in the business intelligence environment, financial fields to do business activities like automated trading, risk management, fraud detection and market forecasting [89–91]. These applications are based on several agents working together to track real-time data streams, make autonomous investment decisions, and react dynamically to changes in market conditions [92,93]. Combining negotiation, reasoning, and learning capabilities, MAS improves the accuracy of the decisions, optimizes the portfolio performance, and increases the resilience of the systems in volatile environments [94].

Li et al. (2024) illustrated application of the AutoGen framework in an e-commerce environment of manufacturing to optimize the management of warehouses as well as financial decision making [95]. The Domain specific agents were implemented for monitoring inventory, scheduling stock replenishment, etc., and enabled modular and cooperative execution of tasks in the dynamic workflows. It paid off, in terms of the increase in inventory turnover, the reduction in cost of holding and shipping and total elimination of time poor salesmen have to spend on inventory calls, and overall savings of 21% in the total operating cost. Despite these successes, there are challenges that have been identified such as state synchronization between the distributed agents, computational scale and fault tolerance. Other limitations were integration with enterprise systems and data security issues.

In the same spirit, Quan and Liu (2024) presented InvAgent, an inventory management system that uses a multi-agent framework (AutoGen framework) [96]. InvAgent uses AutoGen conversational architecture to use user and stage specific agents using structured communication without pre-training. Zero-shot learning enables the system to dynamically update its estimates as the demand and supply conditions change whilst still providing transparent and explainable reasoning. Such a design is an example of the adaptability of MAS systems such as AutoGen; it has

 doi:10.20944/preprints202606.0358.v1

been shown to be useful not just in conversational AI, but also in decision-making and supply chain optimization in the real world, which is adaptive.

5.5. Education and Knowledge Management

It is common in education to apply MAS to build intelligent tutoring systems, adaptive learning platforms, and collaborative e-learning environments [97]. The role of the agents in these systems is as tutors, learners and evaluators, which adapt dynamically to the performance of the students by changing instructional material in real-time and altering teaching plans according to the performance of the students [98,99]. MAS supports distributed knowledge sharing, structuring of content and collaborative learning in knowledge management by enabling coordinated interactions among agents [100,101]. These systems have been proved to improve critical thinking, problem-solving abilities, and students' understanding of concepts from multiple perspectives, which surpasses the traditional approach to education [102]. Moreover, by incorporating AI-based MAS in conventional teaching settings, students are better equipped to face challenging professional issues like in other areas, such as law [103].

Dokku et al. (2024) proposed an adaptive learning prototype based on multiple GPT-4o-based agents connected to each other through the AutoGen framework [99]. All the agents had a particular specialty in adaptive learning and could interact in one of the three modes: unconstrained, semi-constrained and fully constrained. The researchers discovered that completely constrained communication enhanced predictability of tasks and sequencing of instructions and constrained the possibility of responding to various inputs of learners. On the other hand, less restrictive communication made it more flexible but even led to some confusion about the role of the agents. These results have indicated the prospect of MAS coordination in facilitating individualized, knowledge based learning and adaptive teaching models in the contemporary school setting.

5.6. Human-AI Collaboration and Conversational Systems

Multi-Agent System (MAS) platforms are central to facilitating the human-AI collaboration in providing solutions to problems through natural language communication, where each agent is assigned clearly defined roles [104,105]. AutoGen and Langroid are examples of frameworks that support conversational agents' plan, reason, and coordination across specialized tasks to accomplish complex and abstract goals [106]. These systems increase productivity by combining autonomous agent intelligence with human direction in areas like customer service, interactive planning, content generation, scene understanding and multi-turn reasoning [107–111].

The most recent example is ChatMotion (Li et al., 2025), a multimodal MAS architecture that can facilitate human-AI interaction in resolving the motion understanding tasks [112]. Unlike conventional instruction-based methods that are used in multimodal LLMs, ChatMotion may enable dynamic and interactive communication between the agents and allow them to break down a complex query and dynamically plan specialized modules to interpret motion. This architecture is more flexible, has situational awareness and user interactivity and it shows promising signs of how MAS architectures can facilitate more natural, intelligent and effective human-AI interactions.

6. Decision Guide for Selecting Multi-Agent System

6.1. Evaluation Criteria

One of the largest decisions is the choice of the proper MAS framework that influences the aspects of scalability, efficiency, and adaptability. The MAS frameworks enable the intelligent agents to cooperate, automate their functions and make autonomous decisions. It is these capabilities that make MAS frameworks important in solving most complex problems. At the same time, a less than optimal option may create bottlenecks in performance, integration difficulties and inefficiencies in workflow.

 doi:10.20944/preprints202606.0358.v1

In this part, we look at the AutoGen, Langroid and MetaGPT which are among the most popular MAS frameworks, to evaluate their suitability in the context of different applications. This will be compared based on some of our significant considerations which include complexity of the tasks to be performed, real time processing, domain specific tailoring, flexibility of integration and the extent of human interaction that is needed. In this way, the developers and the researchers will be able to make more informed decisions regarding their technical and operational requirements considering the weaknesses and strengths of each framework.

6.2. Framework Recommendations

Analysis of next-generation MAS frameworks suggests that every framework is optimal in specific application situations. AutoGen is effective in those environments that need tool integration, external API connectivity, and interactive workflows that involve human and agent involvement. It works well particularly with automating tasks, debugging, and creating cross-language systems, specifically in cases where asynchronous communication and detailed logging of activities are required. Langroid works well in natural language processing and in tasks that require a lot of knowledge, like chatbots, document analysis, and retrieval-augmented generation systems. The design is lightweight and efficient, giving it an opportunity to be integrated into distributed environments such as IoT and mobile platforms and to support the usage of vector databases and graph-based knowledge structures. MetaGPT is most suitable in applications where there are coordinated, multi-stage workflows, and high-level reasoning, which utilizes role specialization and task decomposition to solve highly-structured projects in the fields of software engineering, research-based problem solving, healthcare systems, and strategic planning. The fact that it yields consistent and standardized outputs also renders it useful especially when it comes to high-stakes situations that are critical to decision-making. Altogether, AutoGen would best fit flexible automation and integration, Langroid is better suited to applications that require light NLP-driven processing, and MetaGPT would be ideal in terms of structured collaboration at scale. A flowchart of these recommendations is given in Figure 12 to be used practically.

 doi:10.20944/preprints202606.0358.v1

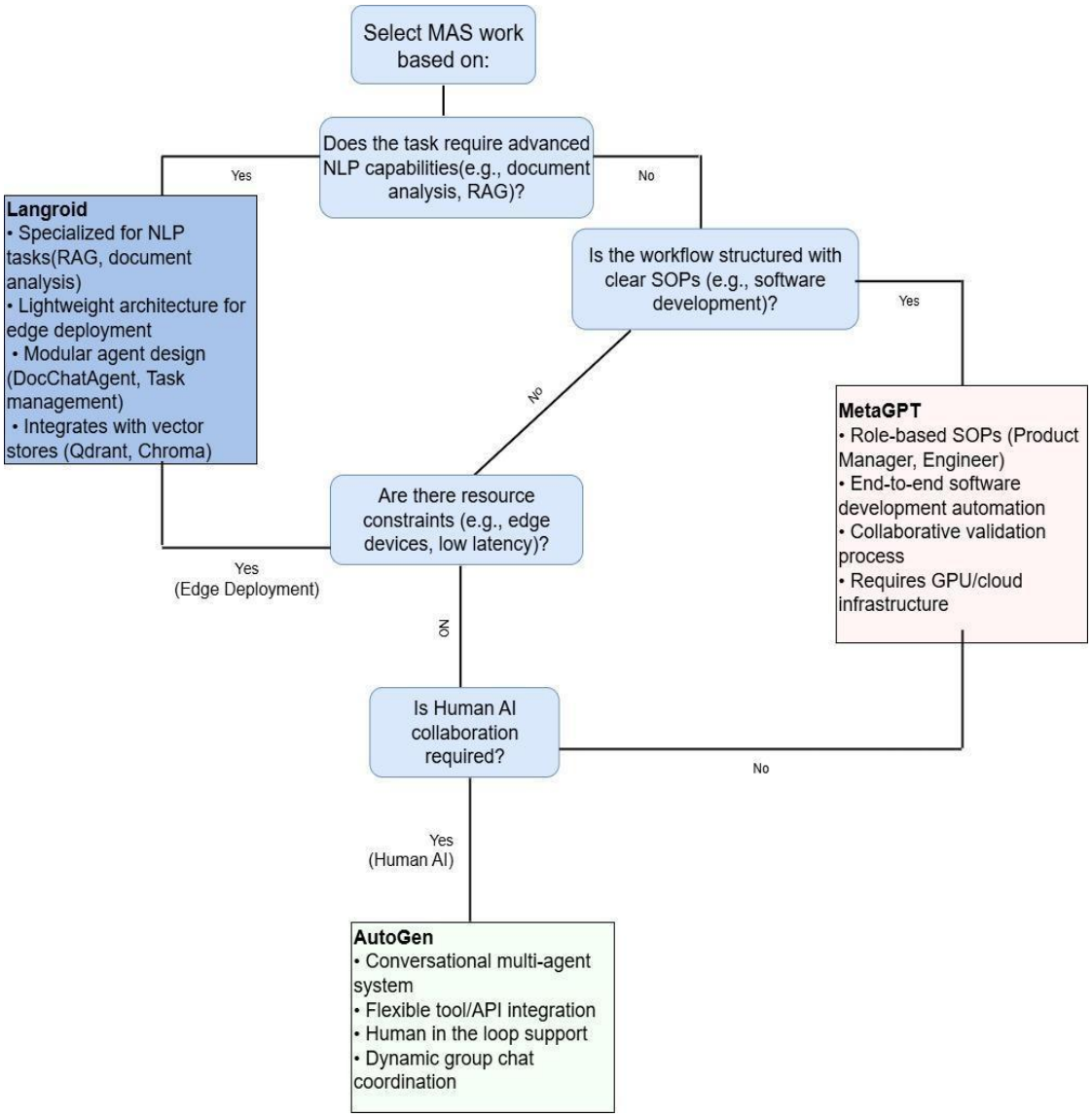


Figure 12. Decision guides for Next-generation MAS frameworks.

Table 4. Framework Selection Criteria for MAS.

Feature	AutoGen	Langroid	MetaGPT
Use Cases	ChatGPT-like conversational agents, research workflows, inventory systems	NLP-focused multi-agent collaboration	Complex reasoning and structured workflows
Scalability	Appropriate for small to medium sized projects (<100 agents)	Efficient for text-based applications, can be used in resource constrained environments	Optimized for large-scale multi-agent workflows
Integration & Extensibility	Strong API support, integrates with external tools (e.g., REST, OpenTelemetry) (Jin et al., 2025) .	Seamless integration with NLP frameworks like SpaCy and Hugging Face and RAG pipelines.	Requires GPU/cloud infrastructure for high-performance computing

 doi:10.20944/preprints202606.0358.v1

Adaptability & Learning	Supports continual learning and evolving AI capabilities	Modular NLP pipelines allow flexible customization	Dynamic agent recruitment and task allocation (Korol, 2023)
Human Involvement	Allows for human in the loop interaction within agent workflows.(Jin et al., 2025)	Mainly autonomous but can involve humans in text-based tasks.	Stores and encodes SOPs for workflows (Jin et al., 2025).
Observability & debugging	Comprehensive logging, cost tracking, and OpenTelemetry support (Cihon et al., 2025) .	Built-in tools for RAG applications (Alhanahnah & Boshmaf, 2024)	Guarantees that the outputs are clear and unambiguous due to the structured nature of the process.
Limitations	High token cost, no open-source models, complex state management, limited reasoning explainability	Primarily focused on NLP; may not be suitable for non-text-based tasks, debugging difficulty, scalability limits	Very computationally intensive, Rigid SOPs, low adaptability
Real world examples	Inventory management in e-commerce.	Pharmacovigilance system (MALADE) (*MALADE: Multi-Agent Architecture for Pharmacovigilance - Langroid*, n.d.)	Software development lifecycle with QA loops (Hong et al., 2024b)

7. Future Directions

The future of MAS frameworks depends on advances in the field of AI, computer infrastructure, and human-machine interaction. While applications of MAS are growing in terms of their scale and complexity, a new direction in research is necessary for addressing not only future challenges but also for further improvements of their performance.

Some of the future research areas involve the integration of Large Language Models into MAS, enabling agents to reason, decide, and coordinate in natural language. Future research should be directed toward allowing communication whereby agents are able to understand context, negotiate, and cooperate with efficiency. Other uses are real-time learning from feedback and improving the efficiency of inferences by using various techniques including pruning, retrieval-augmented generation, and model distillation. Such novel approaches can reduce computational costs while retaining performance.

Another important area of consideration is the expansion of MAS to operate in resource-constrained environments, including the IoT and edge computing. In this context, lightweight decentralized agent frameworks that can be executed on low-capacity devices should be developed. Energy-constrained designs for battery-powered systems and secure designs that maintain user privacy using decentralized decision-making and federated learning must also be explored.

Similarly, human-AI collaboration will lie at the core in the future of MAS infrastructures. Future development should focus on higher explainability and transparency for user trust and understanding of agents' behavior. It will also have to investigate multimodal communication by way of speech, gestures, and augmented reality toward natural and intuitive interaction. Moreover, the area of mixed-initiative systems, where control can shift between humans and AI flexibly with each situation, also needs development. Long-term improvement will finally rely on the development of strong benchmarks and test metrics. Common metrics should be developed that quantify

 doi:10.20944/preprints202606.0358.v1

scalability, responsiveness, and latency across platforms. Domain-specific benchmarks in areas such as robotics, medicine, and finance can be further employed to quantify real-world performance.

Open datasets and mass-scale simulations will allow for honest comparisons and drive innovations. These areas of research together underpin the development of scalable, efficient, secure, and human-centered MAS frameworks with the capability to solve real-world problems in AI and distributed systems.

8. Conclusions

This survey examined recent trends in next-generation MAS systems, including Langroid, AutoGen, and MetaGPT. All these frameworks are significant milestones towards the achievement of successful collaboration between agents that are driven by the use of the LLM, human input, and external tools. We demonstrated through their features, their respective strengths and weaknesses that these frameworks vary in the design objectives and in real-world applications.

In our comparison, we pointed out that the three frameworks are all oriented at making the deployment and coordination of MAS simpler, but are of different nature. To illustrate, AutoGen works well when used to develop conversational software or software that must be powered through reasoning; Langroid can be extremely flexible when it comes to designing and developing modular and customizable MAS designs; and MetaGPT is highly competent with respect to offering structured collaboration for complex task domains.

According to these results, we can suggest a choice of a framework based on an exact set of requirements of the application. In the case of chatbots where virtual assistants and tasks involving dialogues are required, one can apply AutoGen. In situations where the adaptability and modularity are at risk, Langroid can be used. MetaGPT will be used in large-scale collaborative tasks that have the task decomposition and roles assignment as the biggest issues.

Overall, the next generation MAS frameworks are making the multi-agent collaboration more realistic, scalable, and flexible in addressing real-world problems. Among other things, as the research progresses, we can look forward to the additional development into frameworks encompassing more advanced reasoning and efficient computation, as well as human-AI collaboration to open new possibilities for intelligent systems.

References

1. A. Dorri, S. S. Kanhere, and R. Jurdak, “Multi-Agent Systems: A Survey,” *IEEE Access*, vol. 6, pp. 28573–28593, 2018, doi: 10.1109/ACCESS.2018.2831228.
2. P. Stone and M. Veloso, “Multiagent Systems: A Survey from a Machine Learning Perspective,” *Auton. Robots*, vol. 8, no. 3, pp. 345–383, June 2000, doi: 10.1023/A:1008942012299.
3. I. P. Nweke, C. O. Ogadah, K. Koshechkin, and P. M. Oluwasegun, “Multi-Agent AI Systems in Healthcare: A Systematic Review Enhancing Clinical Decision-Making,” *Asian J. Med. Princ. Clin. Pract.*, vol. 8, no. 1, pp. 273–285, May 2025, doi: 10.9734/ajmpcp/2025/v8i1288.
4. S. Park and V. Sugumaran, “Designing multi-agent systems: a framework and application,” *Expert Syst. Appl.*, vol. 28, no. 2, pp. 259–271, Feb. 2005, doi: 10.1016/j.eswa.2004.10.006.
5. W. X. Zhao et al., “A Survey of Large Language Models,” Mar. 11, 2025, *arXiv*: arXiv:2303.18223. doi: 10.48550/arXiv.2303.18223.
6. OpenAI et al., “GPT-4 Technical Report,” Mar. 04, 2024, *arXiv*: arXiv:2303.08774. doi: 10.48550/arXiv.2303.08774.
7. T. Guo et al., “Large Language Model based Multi-Agents: A Survey of Progress and Challenges,” Apr. 19, 2024, *arXiv*: arXiv:2402.01680. doi: 10.48550/arXiv.2402.01680.
8. R. Ye et al., “MAS-GPT: Training LLMs to Build LLM-based Multi-Agent Systems,” Mar. 05, 2025, *arXiv*: arXiv:2503.03686. doi: 10.48550/arXiv.2503.03686.

 doi:10.20944/preprints202606.0358.v1

9. F. Bouquet, S. Chipeaux, C. Lang, N. Marilleau, J.-M. Nicod, and P. Taillandier, "1 - Introduction to the Agent Approach," in *Agent-based Spatial Simulation with Netlogo*, A. Banos, C. Lang, and N. Marilleau, Eds., Oxford: Elsevier, 2015, pp. 1–28. doi: 10.1016/B978-1-78548-055-3.50001-0.
10. J. Xie and C.-C. Liu, "Multi-agent systems and their applications," *J. Int. Counc. Electr. Eng.*, vol. 7, no. 1, pp. 188–197, Jan. 2017, doi: 10.1080/22348972.2017.1348890.
11. T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Interspeech 2010*, ISCA, Sept. 2010, pp. 1045–1048. doi: 10.21437/Interspeech.2010-343.
12. Y. Wu et al., "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," Oct. 08, 2016, *arXiv*: arXiv:1609.08144. doi: 10.48550/arXiv.1609.08144.
13. M. V. Koroteev, "BERT: A Review of Applications in Natural Language Processing and Understanding," Mar. 22, 2021, *arXiv*: arXiv:2103.11943. doi: 10.48550/arXiv.2103.11943.
14. L. Ouyang et al., "Training language models to follow instructions with human feedback," Mar. 04, 2022, *arXiv*: arXiv:2203.02155. doi: 10.48550/arXiv.2203.02155.
15. H. Naveed et al., "A Comprehensive Overview of Large Language Models." 2024. [Online]. Available: https://arxiv.org/abs/2307.06435
16. P. Zhao, Z. Jin, and N. Cheng, "An In-depth Survey of Large Language Model-based Artificial Intelligence Agents." 2023. [Online]. Available: https://arxiv.org/abs/2309.14365
17. A. Hughes, "AutoGen: Enabling next-generation large language model applications," *Microsoft Research*. Sept. 2023. Accessed: May 02, 2025. [Online]. Available: https://www.microsoft.com/en-us/research/blog/autogen-enabling-next-generation-large-language-model-applications/
18. S. Hong et al., "MetaGPT: Meta Programming for Multi-Agent Collaborative Framework." arXiv, Aug. 2023. doi: 10.48550/arXiv.2308.00352.
19. LanceDB, "Langroid: Multi-Agent Programming framework for LLMs," *LanceDB's Substack*. Jan. 2024. Accessed: May 02, 2025. [Online]. Available: https://lancedb.substack.com/p/langoid-multi-agent-programming-framework
20. A. khan et al., "Advances in LLMs with Focus on Reasoning, Adaptability, Efficiency and Ethics," June 14, 2025, *arXiv*: arXiv:2506.12365. doi: 10.48550/arXiv.2506.12365.
21. A. Kantamneni, L. E. Brown, G. Parker, and W. W. Weaver, "Survey of multi-agent systems for microgrid control," *Eng. Appl. Artif. Intell.*, vol. 45, pp. 192–203, Oct. 2015, doi: 10.1016/j.engappai.2015.07.005.
22. R. H. Bordini and J. F. Hübner, "BDI Agent Programming in AgentSpeak Using Jason," in *Computational Logic in Multi-Agent Systems*, F. Toni and P. Torroni, Eds., Berlin, Heidelberg: Springer, 2006, pp. 143–164. doi: 10.1007/11750734_9.
23. P. F. Oliveira, P. Novais, and P. Matos, "Using Jason Framework to Develop a Multi-agent System to Manage Users and Spaces in an Adaptive Environment System," in *Ambient Intelligence – Software and Applications*, P. Novais, G. Vercelli, J. L. Larriba-Pey, F. Herrera, and P. Chamoso, Eds., Cham: Springer International Publishing, 2021, pp. 137–145. doi: 10.1007/978-3-030-58356-9_14.
24. F. Bergenti, E. Iotti, S. Monica, and A. Poggi, "Agent-oriented model-driven development for JADE with the JADEL programming language," *Comput. Lang. Syst. Struct.*, vol. 50, pp. 142–158, Dec. 2017, doi: 10.1016/j.cl.2017.06.001.
25. J. H. Lee and S. C. Park, "Agent and data mining based decision support system and its adaptation to a new customer-centric electronic commerce," *Expert Syst. Appl.*, vol. 25, no. 4, pp. 619–635, Nov. 2003, doi: 10.1016/S0957-4174(03)00101-5.
26. H. Derouiche, Z. Brahmi, and H. Mazeni, "Agentic AI Frameworks: Architectures, Protocols, and Design Challenges," Aug. 13, 2025, *arXiv*: arXiv:2508.10146. doi: 10.48550/arXiv.2508.10146.
27. A. L. Symeonidis, D. D. Kehagias, and P. A. Mitkas, "Intelligent policy recommendations on enterprise resource planning by the use of agent technology and data mining techniques," *Expert Syst. Appl.*, vol. 25, no. 4, pp. 589–602, Nov. 2003, doi: 10.1016/S0957-4174(03)00099-X.
28. H. S. Yim, H. J. Ahn, J. W. Kim, and S. J. Park, "Agent-based adaptive travel planning system in peak seasons," *Expert Syst. Appl.*, vol. 27, no. 2, pp. 211–222, Aug. 2004, doi: 10.1016/j.eswa.2004.01.004.

 doi:10.20944/preprints202606.0358.v1

29. J. M. Solanki, S. Khushalani, and N. N. Schulz, "A Multi-Agent Solution to Distribution Systems Restoration," *IEEE Trans. Power Syst.*, vol. 22, no. 3, pp. 1026–1034, Aug. 2007, doi: 10.1109/TPWRS.2007.901280.
30. A. Sujil, J. Verma, and R. Kumar, "Multi agent system: concepts, platforms and applications in power systems," *Artif. Intell. Rev.*, vol. 49, no. 2, pp. 153–182, Feb. 2018, doi: 10.1007/s10462-016-9520-8.
31. B. M. Radhakrishnan and D. Srinivasan, "A multi-agent based distributed energy management scheme for smart grid applications," *Energy*, vol. 103, pp. 192–204, May 2016, doi: 10.1016/j.energy.2016.02.117.
32. A. Sharma, D. Srinivasan, and D. S. Kumar, "A comparative analysis of centralized and decentralized multi-agent architecture for service restoration," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, July 2016, pp. 311–318. doi: 10.1109/CEC.2016.7743810.
33. M. Li, A. Polyakov, and G. Zheng, "On Generalized Homogeneous Leader-Following Consensus Control for Multiagent Systems," *IEEE Trans. Control Netw. Syst.*, vol. 11, no. 1, pp. 558–568, Mar. 2024, doi: 10.1109/TCNS.2023.3290429.
34. M. Georgiev, I. Tanev, and K. Shimohara, "Performance Analysis and Comparison on Heterogeneous and Homogeneous Multi-Agent Societies in Correlation to Their Average Capabilities," in *2018 57th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, Sept. 2018, pp. 674–679. doi: 10.23919/SICE.2018.8492713.
35. R. Ye et al., "X-MAS: Towards Building Multi-Agent Systems with Heterogeneous LLMs," May 22, 2025, *arXiv*: arXiv:2505.16997. doi: 10.48550/arXiv.2505.16997.
36. P. Chen, S. Liu, B. Chen, and L. Yu, "Multi-Agent Reinforcement Learning for Decentralized Resilient Secondary Control of Energy Storage Systems Against DoS Attacks," *IEEE Trans. Smart Grid*, vol. 13, no. 3, pp. 1739–1750, May 2022, doi: 10.1109/TSG.2022.3142087.
37. D. J. Moore, "A Taxonomy of Hierarchical Multi-Agent Systems: Design Patterns, Coordination Mechanisms, and Industrial Applications," Aug. 18, 2025, *arXiv*: arXiv:2508.12683. doi: 10.48550/arXiv.2508.12683.
38. H. Derouiche, Z. Brahmi, and H. Mazeni, "Agentic AI Frameworks: Architectures, Protocols, and Design Challenges," Aug. 13, 2025, *arXiv*: arXiv:2508.10146. doi: 10.48550/arXiv.2508.10146.
39. "Beyond Self-Talk: A Communication-Centric Survey of LLM-Based Multi-Agent Systems." Accessed: Oct. 04, 2025. [Online]. Available: https://arxiv.org/html/2502.14321v1#bib.bib21
40. C. Lee et al., "A Unified Debugging Approach via LLM-Based Multi-Agent Synergy," Oct. 23, 2024, *arXiv*: arXiv:2404.17153. doi: 10.48550/arXiv.2404.17153.
41. Z. Tang et al., "Towards CausalGPT: A Multi-Agent Approach for Faithful Knowledge Reasoning via Promoting Causal Consistency in LLMs," Feb. 12, 2025, *arXiv*: arXiv:2308.11914. doi: 10.48550/arXiv.2308.11914.
42. E. German and L. Sheremetov, "An Agent Framework for Processing FIPA-ACL Messages Based on Interaction Models," in *Agent-Oriented Software Engineering VIII*, Springer, Berlin, Heidelberg, 2008, pp. 88–102. doi: 10.1007/978-3-540-79488-2_7.
43. J. Pitt and A. Mamdani, "Designing Agent Communication Languages for Multi-agent Systems," in *Multi-Agent System Engineering*, F. J. Garijo and M. Boman, Eds., Berlin, Heidelberg: Springer, 1999, pp. 102–114. doi: 10.1007/3-540-48437-X_9.
44. D. Keil and D. Goldin, "Indirect Interaction in Environments for Multi-agent Systems," in *Environments for Multi-Agent Systems II*, D. Weyns, H. Van Dyke Parunak, and F. Michel, Eds., Berlin, Heidelberg: Springer, 2006, pp. 68–87. doi: 10.1007/11678809_5.
45. "Multi-Agent Coordination across Diverse Applications: A Survey." Accessed: Oct. 04, 2025. [Online]. Available: https://arxiv.org/html/2502.14743v2
46. I. Seilonen, K. Koskinen, T. Pirttioja, P. Appelqvist, and A. Halme, "Reactive and deliberative control and cooperation in multi-agent system based process automation," in *2005 International Symposium on Computational Intelligence in Robotics and Automation*, June 2005, pp. 469–474. doi: 10.1109/CIRA.2005.1554321.

 doi:10.20944/preprints202606.0358.v1

47. O. Simonin and F. Gechter, "An Environment-Based Methodology to Design Reactive Multi-agent Systems for Problem Solving," in *Environments for Multi-Agent Systems II*, D. Weyns, H. Van Dyke Parunak, and F. Michel, Eds., Berlin, Heidelberg: Springer, 2006, pp. 32–49. doi: 10.1007/11678809_3.
48. S. Sharma, "An Overview of Multi Agent Frameworks: Autogen, CrewAI and LangGraph." Apr. 2024. Accessed: May 07, 2025. [Online]. Available: https://sajalsharma.com/posts/overview-multi-agent-fameworks/
49. A. Ullah et al., "Towards a Decentralised Application-Centric Orchestration Framework in the Cloud-Edge Continuum," Apr. 01, 2025, *arXiv*: arXiv:2504.00761. doi: 10.48550/arXiv.2504.00761.
50. "AgentNet: Decentralized Evolutionary Coordination for LLM-based Multi-Agent Systems." Accessed: Oct. 04, 2025. [Online]. Available: https://arxiv.org/html/2504.00587v1
51. J. Borrego-Díaz and J. Galán Páez, "Knowledge representation for explainable artificial intelligence," *Complex Intell. Syst.*, vol. 8, no. 2, pp. 1579–1601, Apr. 2022, doi: 10.1007/s40747-021-00613-5.
52. "A Survey on Context-Aware Multi-Agent Systems: Techniques, Challenges and Future Directions." Accessed: Oct. 04, 2025. [Online]. Available: https://arxiv.org/html/2402.01968v1
53. H. H. L. C. Monte-Alto, M. Morveli-Espinoza, and C. A. Tacla, "Multi-Agent Systems based on Contextual Defeasible Logic considering Focus," Oct. 01, 2020, *arXiv*: arXiv:2010.00168. doi: 10.48550/arXiv.2010.00168.
54. Y. Huang et al., "ROMAS: A Role-Based Multi-Agent System for Database monitoring and Planning," Dec. 18, 2024, *arXiv*: arXiv:2412.13520. doi: 10.48550/arXiv.2412.13520.
55. J. Ferber, O. Gutknecht, and F. Michel, "From Agents to Organizations: An Organizational View of Multi-agent Systems," in *Agent-Oriented Software Engineering IV*, P. Giorgini, J. P. Müller, and J. Odell, Eds., Berlin, Heidelberg: Springer, 2004, pp. 214–230. doi: 10.1007/978-3-540-24620-6_15.
56. Q. Wu et al., "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation." arXiv, Oct. 2023. doi: 10.48550/arXiv.2308.08155.
57. R. Barbosa, R. Santos, and P. Novais, "Collaborative Problem-Solving with LLM: A Multi-agent System Approach to Solve Complex Tasks Using Autogen," in *Highlights in Practical Applications of Agents, Multi-Agent Systems, and Digital Twins: The PAAMS Collection*, A. González-Briones, V. Julian Inglada, A. El Bolock, C. Marco-Detchart, J. Jordan, K. Mason, F. Lopes, and N. Sharaf, Eds., Cham: Springer Nature Switzerland, 2025, pp. 203–214. doi: 10.1007/978-3-031-73058-0_17.
58. S. Hosseini and H. Seilani, "The role of agentic AI in shaping a smart future: A systematic review," *Array*, vol. 26, p. 100399, July 2025, doi: 10.1016/j.array.2025.100399.
59. Gaurav Samdani, Yawal Dixit, and Ganesh Viswanathan, "Leveraging LangGraph and AutoGen for Agentic AI Frameworks," *World J. Adv. Eng. Technol. Sci.*, vol. 8, no. 2, pp. 402–411, Apr. 2023, doi: 10.30574/wjaets.2023.8.2.0068.
60. S. Joshi, " Review of autonomous systems and collaborative AI agent frameworks
 ," Feb. 17, 2025, *Social Science Research Network, Rochester, NY*: 5142205. Accessed: Oct. 06, 2025. [Online]. Available: https://papers.ssrn.com/abstract=5142205
61. H. Wang, J. Gong, H. Zhang, J. Xu, and Z. Wang, "AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities," Sept. 15, 2025, *arXiv*: arXiv:2508.11126. doi: 10.48550/arXiv.2508.11126.
62. *langroid/langroid*. (Oct. 05, 2025). Python. Langroid. Accessed: Oct. 06, 2025. [Online]. Available: https://github.com/langroid/langroid
63. "An empirical evaluation of pre-trained large language models for repairing declarative formal specifications | Empirical Software Engineering." Accessed: Oct. 06, 2025. [Online]. Available: https://link.springer.com/article/10.1007/s10664-025-10687-1
64. M. Alhanahnah and Y. Boshmaf, "DepsRAG: Towards Agentic Reasoning and Planning for Software Dependency Management," Oct. 22, 2024, *arXiv*: arXiv:2405.20455. doi: 10.48550/arXiv.2405.20455.
65. futurepedia, "MetaGPT AI Reviews: Use Cases, Pricing & Alternatives," *futurepedia*. Accessed: June 23, 2025. [Online]. Available: https://www.futurepedia.io/tool/metagpt
66. F. Ling, H. Yang, Y. Xiao, and L. Hu, "Meta GPT-Based Agent for Enhanced Phishing Email Detection," in *Proceedings of the 2024 14th International Conference on Communication and Network Security*, in ICCNS '24. New York, NY, USA: Association for Computing Machinery, Feb. 2025, pp. 78–84. doi: 10.1145/3711618.3711619.

 doi:10.20944/preprints202606.0358.v1

67. Y. Zhou, L. Song, B. Wang, and W. Chen, "MetaGPT: Merging Large Language Models Using Model Exclusive Task Arithmetic," June 27, 2024, *arXiv*: arXiv:2406.11385. doi: 10.48550/arXiv.2406.11385.
68. "GitHub - FoundationAgents/MetaGPT: The Multi-Agent Framework: First AI Software Company, Towards Natural Language Programming." Accessed: Oct. 06, 2025. [Online]. Available: https://github.com/FoundationAgents/MetaGPT
69. T. Zeeshan, A. Kumar, S. Pirttikangas, and S. Tarkoma, "Large Language Model Based Multi-Agent System Augmented Complex Event Processing Pipeline for Internet of Multimedia Things." arXiv, Jan. 2025. doi: 10.48550/arXiv.2501.00906.
70. H. Tao et al., "Code Graph Model (CGM): A Graph-Integrated Large Language Model for Repository-Level Software Engineering Tasks," June 23, 2025, *arXiv*: arXiv:2505.16901. doi: 10.48550/arXiv.2505.16901.
71. S. Y. Chin and D. N. K. Why, "Comparative of Multi-Agent System Frameworks: Crewai, Langchain, and Autogen," July 27, 2025, *Social Science Research Network, Rochester, NY*: 5367964. doi: 10.2139/ssrn.5367964.
72. W. Jin, H. Du, B. Zhao, X. Tian, B. Shi, and G. Yang, "A Comprehensive Survey on Multi-Agent Cooperative Decision-Making: Scenarios, Approaches, Challenges and Perspectives." arXiv, Mar. 2025. doi: 10.48550/arXiv.2503.13415.
73. W. Jin, H. Du, B. Zhao, X. Tian, B. Shi, and G. Yang, "A Comprehensive Survey on Multi-Agent Cooperative Decision-Making: Scenarios, Approaches, Challenges and Perspectives," Mar. 17, 2025, *arXiv*: arXiv:2503.13415. doi: 10.48550/arXiv.2503.13415.
74. J. Harper, "AutoGenesisAgent: Self-Generating Multi-Agent Systems for Complex Tasks," Apr. 25, 2024, *arXiv*: arXiv:2404.17017. doi: 10.48550/arXiv.2404.17017.
75. "MetaAgent: Automatically Constructing Multi-Agent Systems Based on Finite State Machines." Accessed: Oct. 04, 2025. [Online]. Available: https://arxiv.org/html/2507.22606v1
76. M. Alhanahnah and Y. Boshmaf, "DepsRAG: Towards Agentic Reasoning and Planning for Software Dependency Management," Oct. 22, 2024, *arXiv*: arXiv:2405.20455. doi: 10.48550/arXiv.2405.20455.
77. J. Choi et al., "MALADE: Orchestration of LLM-powered Agents with Retrieval Augmented Generation for Pharmacovigilance," Aug. 03, 2024, *arXiv*: arXiv:2408.01869. doi: 10.48550/arXiv.2408.01869.
78. X. Hou et al., "Large Language Models for Software Engineering: A Systematic Literature Review," Apr. 10, 2024, *arXiv*: arXiv:2308.10620. doi: 10.48550/arXiv.2308.10620.
79. A. Garcia, V. Silva, C. Chavez, and C. Lucena, "Engineering multi-agent systems with aspects and patterns," *J. Braz. Comput. Soc.*, vol. 8, pp. 57–72, 2002, doi: https://doi.org/10.1590/S0104-65002002000100006.
80. J. He, C. Treude, and D. Lo, "LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision, and the Road Ahead," *ACM Trans Softw Eng Methodol*, vol. 34, no. 5, p. 124:1-124:30, May 2025, doi: 10.1145/3712003.
81. W. Tao, Y. Zhou, Y. Wang, W. Zhang, H. Zhang, and Y. Cheng, "MAGIS: LLM-Based Multi-Agent Framework for GitHub Issue Resolution," June 27, 2024, *arXiv*: arXiv:2403.17927. doi: 10.48550/arXiv.2403.17927.
82. Z. Rasheed, M. Waseem, M. Saari, K. Systä, and P. Abrahamsson, "CodePori: Large Scale Model for Autonomous Software Development by Using Multi-Agents," 2024, doi: 10.48550/arXiv.2402.01411.
83. S. Khanzadeh, "AgentMesh: A Cooperative Multi-Agent Generative AI Framework for Software Development Automation," July 26, 2025, *arXiv*: arXiv:2507.19902. doi: 10.48550/arXiv.2507.19902.
84. J. Liu et al., "Large Language Model-Based Agents for Software Engineering: A Survey," Sept. 04, 2024, *arXiv*: arXiv:2409.02977. doi: 10.48550/arXiv.2409.02977.
85. T. I. Buldakova, A. V. Lantsberg, and S. I. Suyatinov, "Multi-Agent Architecture for Medical Diagnostic Systems," in *2019 1st International Conference on Control Systems, Mathematical Modelling, Automation and Energy Efficiency (SUMMA)*, Nov. 2019, pp. 344–348. doi: 10.1109/SUMMA48161.2019.8947489.
86. E. Shakshuki and M. Reid, "Multi-Agent System Applications in Healthcare: Current Technology and Future Roadmap," *Procedia Comput. Sci.*, vol. 52, pp. 252–261, Jan. 2015, doi: 10.1016/j.procs.2015.05.071.
87. J. S. Dhatterwal, M. Singh Naruka, and K. S. Kaswan, "Multi-Agent System based Medical Diagnosis Using Particle Swarm Optimization in Healthcare," in *2023 International Conference on Artificial Intelligence and Smart Communication (AISC)*, Jan. 2023, pp. 889–893. doi: 10.1109/AISC56616.2023.10085654.

 doi:10.20944/preprints202606.0358.v1

88. M. Humayun, N. Z. Jhanjhi, A. Almotilag, and M. F. Almufareh, "Agent-Based Medical Health Monitoring System," *Sensors*, vol. 22, no. 8, p. 2820, Jan. 2022, doi: 10.3390/s22082820.
89. "CRMAgent: A Multi-Agent LLM System for E-Commerce CRM Message Template Generation." Accessed: Oct. 07, 2025. [Online]. Available: https://arxiv.org/html/2507.08325v1
90. S. Fatemi and Y. Hu, "FinVision: A Multi-Agent Framework for Stock Market Prediction," in *Proceedings of the 5th ACM International Conference on AI in Finance*, in ICAIF '24. New York, NY, USA: Association for Computing Machinery, Nov. 2024, pp. 582–590. doi: 10.1145/3677052.3698688.
91. J. Bajo, M. L. Borrajo, J. F. De Paz, J. M. Corchado, and M. A. Pellicer, "A multi-agent system for web-based risk management in small and medium business," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 6921–6931, June 2012, doi: 10.1016/j.eswa.2012.01.001.
92. T. F. Mabrouk, M. M. El-Sherbiny, S. K. Guirguis, and A. Y. Shawky, "A Multi-Agent Role-Based System for Business Intelligence," in *Innovations and Advances in Computer Sciences and Engineering*, T. Sobh, Ed., Dordrecht: Springer Netherlands, 2010, pp. 203–208. doi: 10.1007/978-90-481-3658-2_35.
93. S. Wang, Y. Zhao, X. Hou, and H. Wang, "Large Language Model Supply Chain: A Research Agenda," Nov. 26, 2024, *arXiv*: arXiv:2404.12736. doi: 10.48550/arXiv.2404.12736.
94. J. A. García Coria, J. A. Castellanos-Garzón, and J. M. Corchado, "Intelligent business processes composition based on multi-agent systems," *Expert Syst. Appl.*, vol. 41, no. 4, Part 1, pp. 1189–1205, Mar. 2014, doi: 10.1016/j.eswa.2013.08.003.
95. Z. Li, A. Ksibi, and X. Xu, "Optimizing inventory management using a multi-agent LLM system," 2024.
96. Y. Quan and Z. Liu, "InvAgent: A Large Language Model based Multi-Agent System for Inventory Management in Supply Chains," Jan. 31, 2025, *arXiv*: arXiv:2407.11384. doi: 10.48550/arXiv.2407.11384.
97. S. J. Shi, Y. B. Cao, Z. L. Shi, J. W. Li, and R. Zhang, "Application of multi-agent systems in legal education: the impact of multi-agent mock trial exercises on student satisfaction, core skill enhancement, and cognitive development," *Interact. Learn. Environ.*, vol. 0, no. 0, pp. 1–22, doi: 10.1080/10494820.2025.2542892.
98. P. Lagakis and S. Demetriadis, "EvaAI: A Multi-agent Framework Leveraging Large Language Models for Enhanced Automated Grading," in *Generative Intelligence and Intelligent Tutoring Systems*, A. Sifaleras and F. Lin, Eds., Cham: Springer Nature Switzerland, 2024, pp. 378–385. doi: 10.1007/978-3-031-63028-6_32.
99. S. Dokku et al., "MULTI-AGENT ADAPTIVE LEARNING FOR MATHEMATICS".
100. S. Ni and M. Yang, "Educational-Psychological Dialogue Robot Based on Multi-agent Collaboration," in *Social Robotics*, H. Li, T. Schultz, Y. Bi, J. Zhu, H. He, J. Ma, S. Cai, W. Jiang, and S. S. Ge, Eds., Singapore: Springer Nature, 2025, pp. 119–125. doi: 10.1007/978-981-96-1151-5_12.
101. Y.-H. Jiang, T.-Y. Liu, X. Zhuang, H. Hu, R. Li, and R. Jia, "Enhancing Educational Practices with Multi-Agent Systems: A Review".
102. Q. Li, Y. Xie, S. Chakravarty, and D. Lee, "EduMAS: A Novel LLM-Powered Multi-Agent Framework for Educational Support," in *2024 IEEE International Conference on Big Data (BigData)*, Dec. 2024, pp. 8309–8316. doi: 10.1109/BigData62323.2024.10826103.
103. W. Yonghe, J. Yuan-Hao, C. Yuanyuan, and Z. Wenxuan, "Multi-Agent Systems Supported by Large Language Models: Technical Pathways, Educational Applications, and Future Prospects".
104. W. Chen et al., "AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors," Oct. 23, 2023, *arXiv*: arXiv:2308.10848. doi: 10.48550/arXiv.2308.10848.
105. R. Gody, M. Goudy, and A. Y. Tawfik, "ConvoGen: Enhancing Conversational AI with Synthetic Data: A Multi-Agent Approach," May 09, 2025, *arXiv*: arXiv:2503.17460. doi: 10.48550/arXiv.2503.17460.
106. N. Kugo et al., "VideoMultiAgents: A Multi-Agent Framework for Video Question Answering," Apr. 30, 2025, *arXiv*: arXiv:2504.20091. doi: 10.48550/arXiv.2504.20091.
107. C. Wan, "AI Agent-Enhanced Navigation and Perception : A Survey of Strategies in Virtual Environments." Accessed: Oct. 07, 2025. [Online]. Available: http://www.theseus.fi/handle/10024/896597
108. H. Kim, K. Mitra, C. Shen, D. Zhang, and E. Hruschka, "AIPOM: Agent-aware Interactive Planning for Multi-Agent Systems," Sept. 29, 2025, *arXiv*: arXiv:2509.24826. doi: 10.48550/arXiv.2509.24826.
109. J. Li, P. Huang, Y. Li, S. Chen, J. Hu, and Y. Tian, "A Unified Multi-Agent Framework for Universal Multimodal Understanding and Generation," Aug. 14, 2025, *arXiv*: arXiv:2508.10494. doi: 10.48550/arXiv.2508.10494.

 doi:10.20944/preprints202606.0358.v1

110. Y. Fan et al., "VideoAgent: A Memory-Augmented Multimodal Agent for Video Understanding," in *Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds., Cham: Springer Nature Switzerland, 2025, pp. 75–92. doi: 10.1007/978-3-031-72670-5_5.
111. J. Rao, Z. Li, H. Wu, Y. Zhang, Y. Wang, and W. Xie, "Multi-Agent System for Comprehensive Soccer Understanding," Sept. 02, 2025, *arXiv*: arXiv:2505.03735. doi: 10.48550/arXiv.2505.03735.
112. L. Li et al., "ChatMotion: A Multimodal Multi-Agent for Human Motion Analysis," Feb. 27, 2025, *arXiv*: arXiv:2502.18180. doi: 10.48550/arXiv.2502.18180.
113. [113] Y. Liu, J. Cai, Y. Li, et al., "MASFactory: A Graph-centric Framework for Orchestrating LLM-Based Multi-Agent Systems with Vibe Graphing," arXiv preprint arXiv:2603.06007, Mar. 2026
114. G. Liu, H. Lin, H. Zeng, et al., "MAS-on-the-Fly: Dynamic Adaptation of LLM-based Multi-Agent Systems at Test Time," arXiv preprint arXiv:2602.13671, Feb. 2026.
115. C. Paduraru, P.-L. Bouruc, and A. Stefanescu, "A Trace-Based Assurance Framework for Agentic AI Orchestration: Contracts, Testing, and Governance," arXiv preprint arXiv:2603.18096, Mar. 2026.
116. A. Khan, A. Zainab, S. H. Khan, A. Ishaq, and H. Asif, "Emergent Intelligence in Multi-Agent and LLM Systems: A Survey and Perspective Toward Autonomous, Collaborative, and Generalizable AI," TechRxiv preprint, techrxiv.177092236.62657640 Feb 2026.