A Taxonomy of Architecture Options for Foundation Model-based Agents: Analysis and Decision Model

Jingwen Zhou
CSIRO's Data61
Melbourne, Australia
helen.zhou@data61.csiro.au

Qinghu Lu
CSIRO's Data61
University of New South Wales
Sydney, Australia
qinghua.lu@data61.csiro.au

Jieshan Chen
CSIRO's Data61
Sydney, Australia
jieshan.chen@data61.csiro.au

Liming Zhu
CSIRO's Data61
University of New South Wales
Sydney, Australia
liming.zhu@data61.csiro.au

Xiwei Xu
CSIRO's Data61
Sydney, Australia
xiwei.xu@data61.csiro.au

Zhenchang Xing
CSIRO's Data6
Australian National University
Canberra, Australia
zhenchang.xing@data61.csiro.au

Stefan Harrer
CSIRO's Data61
Sydney, Australia
stefan.harrer@data61.csiro.au

Abstract—The rapid advancement of AI technology has led to widespread applications of agent systems across various domains. However, the need for detailed architecture design poses significant challenges in designing and operating these systems. This paper introduces a taxonomy focused on the architectures of foundation-model-based agents, addressing critical aspects such as functional capabilities and non-functional qualities. We also discuss the operations involved in both design-time and run-time phases, providing a comprehensive view of architectural design and operational characteristics. By unifying and detailing these classifications, our taxonomy aims to improve the design of foundation-model-based agents. Additionally, the paper establishes a decision model that guides critical design and runtime decisions, offering a structured approach to enhance the development of foundation-model-based agents. Our contributions include providing a structured architecture design option and guiding the development process of foundation-model-based agents, thereby addressing current fragmentation in the field.

Index Terms—Foundation Model, Large Language Model, LLM, Agent, Software Architecture, Taxonomy

I. INTRODUCTION

The rapid advancement of AI technology, particularly foundation models, has led to widespread applications of foundation model based agent systems across various domains, from healthcare and finance to autonomous driving and smart manufacturing [1, 2, 3, 4]. These foundation-model-based agent systems have the potential to revolutionize industries by enhancing efficiencies, enabling automation, and facilitating complex decision-making processes [5].

Recent innovations in AI demonstrate the versatility of foundation-model-based agents. For example, Auto-GPT for autonomous task management and internet searches [1],

BabyAGI has evolved from simple code to include functionalities such as robotics and code-writing [2].

Moreover, existing studies often focus narrowly on specific aspects of AI agents, such as their functional capabilities or performance metrics. For example, the development of GPT4 by OpenAI [6] has shown significant advancements in prompt engineering. Similarly, the integration of retrieval-augmented generation (RAG) [7] has improved the ability of AI systems to generate contextually relevant responses by retrieving information from external databases. Additionally, tools like LangChain and Hugging Face [3,4], which facilitate the integration of multiple AI models and data sources, are often discussed in isolation. These advancements, while significant, are often not considered together in a holistic framework that addresses the overall architectural design and operational characteristics of AI agents. This fragmented approach can result in a lack of comprehensive analysis of agent architecture options.

Despite the innovative application of foundation model-based agents like AutoGen and MetaGPT [8, 9], a significant gap exists in the systematic analysis of their architectural designs. Taxonomies are employed in the software architecture community to deepen the understanding of current technologies [10]. These agents, which harness cutting-edge technologies and introduce new operational paradigms, underscore the necessity for a unified taxonomy that can standardize the classification and elucidate the varying capabilities of such systems. Current frameworks are robust, while often

---

[1] https://github.com/Significant-Gravitas/AutoGPT
[2] https://github.com/yoheinakajima/babyagi
[3] https://www.langchain.com/
[4] https://huggingface.co/

overlooking the critical evaluation of how these architectures can be optimized for better functionality.

Therefore, we propose a comprehensive taxonomy for foundation-model-based agents. Developed through a systematic literature review (SLR), this taxonomy categorizes both the functional capabilities and non-functional qualities of these agents, aiming to serve as a cornerstone in the software architecture community. It offers a deeper understanding and streamlined design options for complex systems, enhancing the design process and facilitating robust comparisons and assessments of design alternatives. Specifically, the taxonomy provides detailed classifications including input modality support, access to underlying models, and integration of external capabilities. This structured approach not only ensures effective agent coordination and communication but also serves as a comprehensive guide for software architects in designing foundation-model-based agent systems. The main contributions of this paper are as follows:

Firstly, it introduces a nuanced taxonomy that categorizes agents based on input modality, access to models, external capabilities, agent coordination, agent communication, etc. This taxonomy analyzes different design options and the trade-offs of quality attributes, providing a comprehensive framework that serves as a definitive guide for designing and enhancing agent-based architectures. Secondly, the paper establishes a decision model that offers a structured approach to guide design decisions. This decision model provides a design guide that enhances the strategic planning and execution of foundation model-based agents, ensuring that critical design and runtime decisions are well-informed.

Specifically, Section II discusses the related work. Section III discusses the methodology. Section IV delves into the detailed taxonomy and design model, elaborating on agent characteristics, capabilities, design-time structures, and runtime operations. Section V introduces the threats to validity. Finally, Section VI concludes the paper, summarizing our key findings and outlining future research directions.

II. BACKGROUND AND RELATED WORK

Recent advancements in foundation-model-based agent systems have seen significant contributions from major tech companies, focusing on enhancing the capabilities and applications of these systems through large language models (LLMs) and innovative multi-agent systems. For instance, Google introduced several AI-driven features at their I/O 2023 event, including the new PaLM2 model, which is optimized for various tasks such as logic, reasoning, and multilingual understanding [11]. This model is integrated into over 25 new products, enhancing their functionality across domains like coding, writing, and mathematics. Google's advancements with Bard [12] and the PaLM 2 model illustrate the potential of LLMs in enhancing agent functionalities [13]. These advancements illustrate how Google is leveraging LLMs to create more sophisticated and versatile AI agents capable of performing diverse tasks effectively.

In addition to Google's contributions, several other architectures and frameworks have emerged as key players in the development of foundation-model-based agents. Meta's advancements with their GenAI infrastructure, including highly efficient and scalable LLMs like LLaMA 2, have set a new standard for AI-driven capabilities [14, 15]. Furthermore, MetaGPT attempts to emulate the structure of traditional software companies by assigning roles such as project managers and engineers to agents, fostering collaborative development of user-defined coding tasks [9]. These developments highlight Meta's focus on building scalable and collaborative AI agent systems. Microsoft has made substantial advancements with its AutoGen framework [8], an open-source system designed to facilitate the communication and collaboration of multiple AI agents, thereby improving performance and reducing errors [16]. Microsoft Copilot integrates AI into various tools, enhancing productivity for different roles, while Azure AI infrastructure supports the scalability of generative AI applications with powerful NVIDIA GPUs. Furthermore, frameworks like CrewAI and LangChain have gained traction for their innovative approaches to agent coordination and workflow integration. CrewAI focuses on enhancing multi-agent collaboration through distributed learning mechanisms [17], while LangChain integrates LLMs with various tools and platforms to enable the seamless execution of complex tasks [18].

Despite significant advancements in foundation-model-based agent systems, a notable gap persists in the comprehensive analysis and standardization of agent architecture options. For example, the AIOS platform integrates LLMs to create a cohesive environment for autonomous agents [19], however, it does not address the need for a standardized approach to defining agent roles and capabilities comprehensively. Similarly, while the MAToM-SNN framework leverages spiking neural networks for enhanced multi-agent cooperation and competition, it too lacks a unified taxonomy that would standardize these systems [20]. Moreover, IBM's initiatives underscore the challenges of integrating LLMs into automation frameworks but fall short in offering a standardized framework that clearly defines agent roles and capabilities [21].

Meanwhile, recent research studies underscore diverse challenges and advancements in LLM-based autonomous agents. One survey highlights the extensive use of LLMs in autonomous agent frameworks, focusing on how these models enhance agent intelligence and functionality but criticizes the lack of standardized architectures necessary for integration [22]. Another study highlights how advancements in prompting techniques enhance model efficiency, yet overlook the need for architectural standardization [23]. Additional work reviews progress and challenges in the field of multi-agent systems enhanced by LLMs, emphasizing the need for a standardized framework to integrate these systems cohesively [24]. Researchers in [25] explore the capabilities of LLMs to simulate human-like decision-making, emphasizing the need for more robust frameworks to support the scalability and
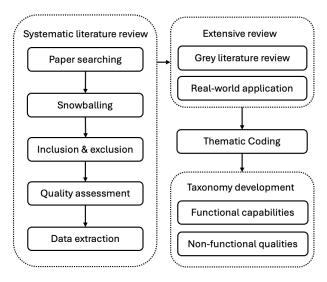
Fig. 1. Methodology

ethical deployment of such technologies. Moreover, the study investigated workflows and components in LLM agents suggesting potential efficiencies but lacking unified architectural integration [26]. These studies point out the necessity for comprehensive design architecture that can evaluate and systematically integrate these technologies into existing and new systems.

Our work aims to fill this gap by offering a detailed taxonomy-based design architecture, ensuring a holistic and systematic approach to integrating these advanced technologies into scalable and ethically responsible systems.

III. METHODOLOGY

Our research study employed a structured and comprehensive approach to address architectural design challenges in systems integrating foundation model-based agents. This methodology can be divided into three main phases: a systematic literature review (SLR), an extensive review, thematic coding, and the development of a taxonomy.

A. Systematic Literature Review (SLR)

Paper Searching: We conducted literature searches in various academic databases and journals using predetermined keywords related to foundation model-based agents. This initial search aimed to gather a broad range of relevant academic publications and scholarly works (300 papers).

Snowballing: To ensure thoroughness, we applied both forward and backward snowballing techniques. Forward snowballing involved examining the citations of initially identified papers, while backward snowballing entailed reviewing references within those papers to identify additional relevant studies (100 papers).

Inclusion & Exclusion Criteria: We established a set of inclusion and exclusion criteria to filter the papers. This step ensured that only studies meeting specific relevance and quality standards were selected for further analysis.

Quality Assessment: The selected papers underwent a rigorous quality assessment process to ensure the robustness and reliability of the included studies. This step was crucial to maintain the integrity of our data extraction (68 studies).

Data Extraction: Finally, we extracted relevant data from the qualifying studies. This phase involved a detailed examination and synthesis of information from 87 studies, which were deemed suitable for our analysis.

B. Extensive Review

Grey Literature Review: Beyond academic publications, we reviewed grey literature to capture the latest trends and applications of foundation model-based agents. This included technical reports, white papers, and other non-peer-reviewed documents (9 studies).

Real-World Application Analysis: To understand practical implementations, we analyzed known real-world applications of foundation model agents. This involved scrutinizing official websites, available documents, and case studies of organizations utilizing these agents (10 studies).

C. Thematic Coding

Our thematic coding process utilized a hybrid approach, combining both deductive and inductive methods to achieve a comprehensive understanding of foundation-model-based agents. This hybrid approach allowed us to systematically categorize the extracted metrics into a structured yet adaptable framework. The predefined criteria provided a clear direction for our analysis, while the emergent sub-criteria offered depth and nuanced insights, resulting in a robust and thorough comprehension of the agents' architecture.

Deductive Coding: We initiated our process with two broad predefined criteria: functional capabilities and non-functional qualities. These criteria guided the initial categorization of the extracted metrics, ensuring a focused and structured analysis.

Emergent Sub-Criteria: As the coding process progressed, sub-themes emerged organically based on patterns, similarities, and differences observed in the data. This inductive method enabled us to capture detailed and context-specific aspects of the metrics, enriching our understanding of their interrelationships and significance.

Refinement and Integration: The emergent sub-criteria were continuously refined and integrated into the overarching predefined themes. This iterative process ensured that our thematic structure accurately reflected the complexities and subtleties of the data, providing a comprehensive framework for analysis.

Internal Validation: To validate our thematic structure, one author conducted the initial coding, followed by a review and feedback process involving six authors. This collaborative validation ensured consensus and accuracy in our categorization.

Upon completing the literature review and thematic coding, our analysis identified twelve key taxonomy branches, systematically categorized under two primary criteria: functional capabilities and non-functional qualities. These branches align

with the overarching pillar criteria depicted in our taxonomy framework.

D. Development of the Taxonomy and the Decision Model

Combining findings from the literature review, and thematic coding, we developed a taxonomy and a decision model of architecture options for foundation model-based agents. This taxonomy focused on two main aspects: functional capabilities and non-functional qualities. To guide our taxonomy and decision model development, we formulated the following research questions (RQs):

RQ1: What are the **key functional capabilities** of foundation-model-based agents?

RQ2: What are the **non-functional qualities** that influence the performance and reliability of foundation-model-based agents?

RQ3: How can a decision model act as a design guide to streamline the development of foundation-model-based agents in complex decision-making environments?

Through this systematic and extensive methodology, we aim to provide clear and structured guidance for the architectural design of foundation model-based agents, thus supporting future research and practical application development in the field.

IV. TAXONOMY OF FOUNDATION-MODEL BASED AGENTS

In this section, we present a taxonomy that defines architecture design options for foundation-model-based agents. The taxonomy is structured into two categories: functional capability (Section IV-A - Section IV-I) and non-functional qualities (Section IV-J). We discuss the characteristics of each design option to consider during the building process and their impact on the foundation-model-based agents.

A. Input Modality

Modality defines whether an agent operates using a single modality or multiple modalities. **Single-modality** agents utilize one type of input such as text, vision, or audio

, making them ideal for straightforward tasks that require less computational resources and simpler data interpretation. For instance, text-based chatbots or vision-only surveillance systems operate within this single modality [27]. In contrast, **multi-modality** agents combine video/audio/image/text inputs uploaded by humans, and understand the operational and environmental context, enabling a more comprehensive and rich interaction with their environment. This enhanced capability allows them to handle more complex tasks like autonomous navigation or interactive virtual assistants that respond not only to voice commands but also to visual and contextual cues. This approach not only facilitates richer user interactions but also aligns with the evolving demands of dynamic environments where adaptability and context awareness are crucial [28].

B. Access to Underlying Models

1) Underlying Model Types: **Non-AI-based agents** operate without employing artificial intelligence techniques. They typically rely on predefined scripts, workflows, or simple automation rules that do not involve learning or adapting over time. Non-AI-based agents are effective in environments where tasks are repetitive and straightforward, as they execute tasks exactly as specified by their programming. While they are fragile and sensitive to the input. An example of a non-AI-based agent is a basic automation script that processes incoming emails and sorts them into different folders based on predefined keywords.

For **AI-based agents**, we consider two types in terms of their purposes: `narrow AI-based agents` and `general purpose AI(GPAI)-based agents`. `Narrow AI-based agents` are designed to perform specific tasks or solve particular problems. These agents use machine learning and other AI techniques to improve their performance in their specialized domains [29]. These agents are effective within their domains but lack generalization capabilities. `GPAI-based agents` aim to perform various tasks across different domains. They are designed with broader capabilities, leveraging extensive datasets and sophisticated algorithms to adapt to multiple environments and challenges [30, 31]. GPAI-based agents can switch between tasks, learn from diverse experiences, and apply their knowledge to new, unforeseen problems, making them more versatile compared to narrow AI agents. This adaptability is achieved through advanced architectures and continuous learning mechanisms, which enable GPAI-based agents to update their models and improve over time.

2) Model Composition: The composition of agent models is crucial for optimizing performance and scalability in agent-based systems. Various configurations can be adopted depending on the complexity and requirements of the system. A **single-model agent** handles all tasks, which is suitable for simpler systems with a limited scope. This approach minimizes complexity but may not scale well for diverse tasks [24, 32].

In more complex systems, **multi-model** configurations are often employed. Multiple models are used to handle different tasks or to enhance performance. One approach is the `mixture of experts`, where different models specialize in different tasks, and the system dynamically selects the appropriate model for each task [33]. Another approach is the `ensemble method`, where multiple models work together to improve accuracy and robustness by combining their outputs [34]. Additionally, `model merging` involves combining different models to create a more capable composite model, such as merging language models and visual recognition models for a comprehensive virtual assistant [35]. A `hybrid model` combines various types of models, such as rule-based and learning-based models, to leverage their respective strengths. This approach is used in cybersecurity systems that employ rule-based detection for known threats and machine learning models for anomaly detection [36].
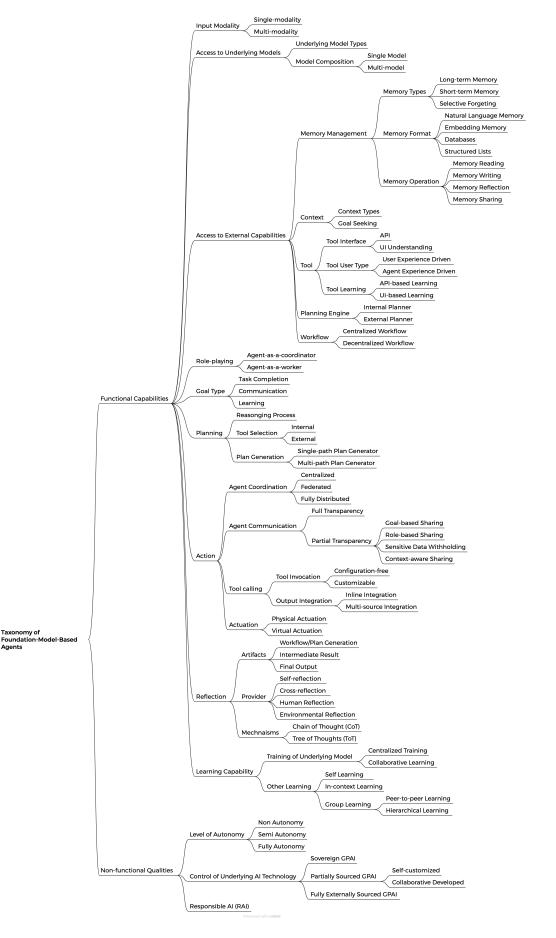
Taxonomy of Foundation-Model-Based Agents

- Functional Capabilities
  - Input Modality
    - Single-modality
    - Multi-modality
  - Access to Underlying Models
    - Underlying Model Types
    - Model Composition
      - Single Model
      - Multi-model
  - Access to External Capabilities
    - Memory Management
      - Memory Types
        - Long-term Memory
        - Short-term Memory
        - Selective Forgetting
      - Memory Format
        - Natural Language Memory
        - Embedding Memory
        - Databases
        - Structured Lists
      - Memory Operation
        - Memory Reading
        - Memory Writing
        - Memory Reflection
        - Memory Sharing
    - Context
      - Context Types
      - Goal Seeking
    - Tool
      - Tool Interface
        - API
        - UI Understanding
      - Tool User Type
        - User Experience Driven
        - Agent Experience Driven
      - Tool Learning
        - API-based Learning
        - UI-based Learning
    - Planning Engine
      - Internal Planner
      - External Planner
    - Workflow
      - Centralized Workflow
      - Decentralized Workflow
  - Role-playing
    - Agent-as-a-coordinator
    - Agent-as-a-worker
  - Goal Type
    - Task Completion
    - Communication
    - Learning
  - Planning
    - Reasonging Process
    - Tool Selection
      - Internal
      - External
    - Plan Generation
      - Single-path Plan Generator
      - Multi-path Plan Generator
  - Action
    - Agent Coordination
      - Centralized
      - Federated
      - Fully Distributed
    - Agent Communication
      - Full Transparency
      - Partial Transparency
        - Goal-based Sharing
        - Role-based Sharing
        - Sensitive Data Withholding
        - Context-aware Sharing
    - Tool calling
      - Tool Invocation
        - Configuration-free
        - Customizable
      - Output Integration
        - Inline Integration
        - Multi-source Integration
    - Actuation
      - Physical Actuation
      - Virtual Actuation
  - Reflection
    - Artifacts
      - Workflow/Plan Generation
      - Intermediate Result
      - Final Output
    - Provider
      - Self-reflection
      - Cross-reflection
      - Human Reflection
      - Environmental Reflection
    - Mechnaisms
      - Chain of Thought (CoT)
      - Tree of Thoughts (ToT)
  - Learning Capability
    - Training of Underlying Model
      - Centralized Training
      - Collaborative Learning
    - Other Learning
      - Self Learning
      - In-context Learning
      - Group Learning
        - Peer-to-peer Learning
        - Hierarchical Learning
- Non-functional Qualities
  - Level of Autonomy
    - Non Autonomy
    - Semi Autonomy
    - Fully Autonomy
  - Control of Underlying AI Technology
    - Sovereign GPAI
    - Partially Sourced GPAI
      - Self-customized
      - Collaborative Developed
    - Fully Externally Sourced GPAI
  - Responsible AI (RAI)

Presented with xmind

Fig. 2. Taxonomy

C. Access to External Capabilities

1) Memory Management: Memory management is critical for foundation-model-based agents to store, retrieve, and utilize information effectively.

- Memory Types: **Long-term memory** retains information over long periods, which is essential for tasks that require historical data, knowledge, past observations, and past experiences [37, 38]. **Short-term memory**, on the other hand, handles short-term information relevant to immediate tasks [39]. This type of memory is useful for temporary activities that do not require long-term storage, for example, configuration, working context, recent events, and the information within the context window of the FM [40]. **Selective forgetting** enables agents to discard irrelevant or outdated information, ensuring that they maintain optimal performance without being bogged down by unnecessary data [41, 42]. This capability is crucial in dynamic environments where the relevance of data changes rapidly.

- Memory Format: In the foundation-model-based agent systems, several distinctive structures offer unique advantages and serve as pivotal design options depending on specific application needs. **Natural language memory**, as used in systems like Reflexion [43] and Voyager [44], provides a flexible format that stores information in an easily understandable form, facilitating intuitive interaction and preserving rich semantic details to guide agent actions. **Embedding memory**, exemplified by MemoryBank [45] and ChatDev, encapsulates memory information into compact embedding vectors, significantly enhancing the efficiency of memory retrieval processes. **Databases** allow for robust memory manipulation; systems like ChatDB [46] and DB-GPT [47] use databases to enable precise memory operations through SQL queries, providing structured and efficient data management. **Structured lists** are employed in models such as GITM [48] and RET-LLM [49], where memory is systematically organized in lists or hierarchical structures that clearly define the relationships between elements and facilitate rapid data access. Each format presents distinct advantages: natural language for clarity and semantic richness, embeddings for retrieval speed, databases for structured manipulation, and lists for organized and hierarchical memory storage. These memory formats can be integrated, as demonstrated by GITM, which uses a hybrid approach combining key-value pairs with embeddings and natural language to maximize retrieval efficiency and content comprehensiveness, providing multiple design options to optimize agent performance based on the specific needs of the application.

- Memory Operation: Agents can acquire, accumulate, and utilize knowledge through interactions with their environment via various operations. These operations include memory reading, writing, reflection, and sharing. **Memory reading** involves extracting valuable informa-

tion based on recency, relevance, and importance to enhance the agent's actions [48, 50]. **Memory writing** focuses on storing perceived environmental information while managing duplication and preventing overflow [6]. **Memory reflection** enables agents to summarize and infer high-level insights from past experiences, facilitating more abstract and complex decision-making [50]. **Memory sharing** allows agents to access and contribute to a common memory pool. This enhances their collaborative abilities by integrating diverse experiences and knowledge [38, 51, 52]. In foundation-model-based agent systems, memory sharing involves storing and retrieving shared memories, which supports continuous learning and adaptation, helping agents use the most relevant information for tasks.

2) Context: Context management focuses on gathering and organizing the context within which the agent operates to better understand the user's intentions and goals [27]. There are various of **context types** of information, such as screen recordings [53], mouse clicks, typing patterns, eye tracking, gestures [54], and annotations. These types of context information are crucial for accurately interpreting user goals. When it comes to **goal seeking**, there are two methods: `passive suggestion` and `proactive suggestion`. Passive suggestion analyzes goals explicitly articulated by the user through text prompts submitted via a dialogue interface [55, 56]. In contrast, the proactive suggestion goes beyond explicit text prompts by interpreting the user interface of relevant tools and interactions, using multimodal context information to anticipate user goals [54]. From an architectural perspective, passive suggestions offer straightforward goal interpretation, while proactive suggestions use rich context information for more accurate predictions, providing diverse options for developing adaptable and intelligent systems.

3) Tool:

- Tool Interface: foundation-model-based agents can leverage APIs and UI understanding to interact with external tools and resources. Using **APIs**, these agents can directly call specific functions or retrieve data from other systems, ensuring seamless integration and efficient performance [25, 57, 58, 59]. APIs provide a standardized way for agents to access a wide range of functionalities, from retrieving real-time data to executing complex operations, thus enhancing their adaptability and robustness in various applications. On the other hand, **UI understanding** enables agents to interact with tools and applications through their graphical interfaces [60]. This approach is particularly useful when APIs are unavailable or insufficient [60, 61]. By analyzing visual elements, screen recordings, mouse clicks, and user interactions, agents can comprehend and navigate UIs to achieve their goals. This dual capability of using APIs and UI understanding allows foundation-model-based agents to operate effectively in diverse environments, improving their flexibility and operational efficiency.

- Tool User Type: In the architecture of foundation-model-based agents, the ability to discover and integrate tools is crucial for enhancing adaptability. This involves considering different types of tool users, such as agents themselves or human users. The **user experience-driven tools** are tailored to evolve based on direct user interactions, significantly improving usability and overall user satisfaction [62, 63]. Such advancements are pivotal in shaping how agents interact in user-centric environments. For example, such as those documented by Google, underscore the potential of user feedback to refine AI tools dynamically, making them more intuitive and aligned with user needs [64]. On the other hand, **agent experience driven tools** harness historical data to refine and optimize operational strategies continuously [65, 66]. This methodology is particularly beneficial in environments characterized by variability and change, enhancing the agent's decision-making processes and operational efficiency.
- Tool Learning: Tool learning emphasizes how agents acquire new tool usage capabilities. This process can be categorized into **API-based learning** and **UI-based learning**. In API-based learning, agents learn to use tools by interacting with APIs. This involves understanding and executing programmatic instructions, which enhances their functionality and efficiency [59]. In UI-based learning, agents learn to use tools by reading and interpreting UI interfaces. For instance, UTA (Universal Tool for Agents) exemplifies this approach by traversing UI elements to build a graph and subsequently learning how to use the tools through these interactions [67].

4) Planning Engine: The Planning engine serves as a central component in foundation-model-based agents, orchestrating the high-level planning activities necessary for strategic operations. This engine is crucial for enabling agents to process reasoning, generate plans, monitor their execution, and adapt dynamically to new inputs and environmental shifts, enhancing the agents' effectiveness and adaptability in diverse settings [65, 68, 69]. This can be divided into two primary strategies: **Internal planner** and **external planner**, each addressing different operational needs within the system. Internal planners are embedded within the agent's core architecture and utilize the agent's intrinsic capabilities to autonomously generate and execute plans. External planners incorporate specialized, domain-specific tools that extend the basic functionalities of the planning engine to tackle complex, high-stakes tasks requiring detailed and precise planning [70]. These planners enhance the system's planning capabilities by translating intricate task descriptions into structured action sequences, which are then executed by the agents' foundational models. This not only ensures robust plan formulation but also adapts the execution strategies to meet the specific demands of the tasks.

5) Workflow: Workflow in foundation model-based agents can be organized into two designs: centralized and decentralized workflow. **Centralized workflow** concentrates task management at a single control point, enhancing operational consistency but potentially leading to bottlenecks—a scenario similar to centralized coordination in agent systems (refer to agent coordination section IV-G1). In contrast, **decentralized workflow** spreads task control across multiple agents, increasing system flexibility and resilience, reflecting principles found in distributed coordination strategies. Significantly, workflow is dedicated to the procedural execution of tasks, which is distinct from strategic planning (section IV-F) and the focus on inter-agent interactions characteristic of agent coordination.

D. Role-Playing

In foundation-model-based agent systems, the roles of agents are crucial as they define the functions and interactions within the system. The agents can be categorized into two types: **agent-as-a-coordinator** and **agent-as-a-worker**. Agents in the coordinator role primarily formulate high-level strategies and orchestrate the execution of tasks by delegating task execution responsibilities to other agents, external tools, or non-agent systems, ensuring that tasks are allocated efficiently and that the system operates cohesively [8]. On the other hand, agents in the worker role need to generate strategies and execute specific tasks in line with those strategies [71]. They are the core executors and can be designed to operate autonomously or semi-autonomously based on predefined rules or learning algorithms. To complete these tasks, agents in the worker role may need to cooperate or compete with other agents, or call external tools or non-agent AI/non-AI systems.

E. Goal Type

Agents typically establish goals like **task completion**, **communication**, and **learning**, which then guide the planning and action phases. Task completion goals mean where agents are programmed to achieve specific, complex objectives, such as crafting items in virtual environments like Minecraft [44] or executing specific functions during software development [46]. These tasks are clearly defined with each action strategically aligned towards achieving the outcome. Secondly, communication goals involve agents engaging in interactions with other agents or humans to exchange information or collaborate on joint tasks. For instance, agents within platforms like ChatDev may coordinate efforts in software development [46], while agents like those in Inner Monologue adapt their strategies based on real-time human feedback, showcasing their adaptive communication capabilities [72]. Lastly, learning goals are identified where agents aim to navigate and adapt to unfamiliar settings, balancing between exploration of new areas and exploitation of known resources. An example of this can be seen in agents like Voyager [44], which explore and refine skills through continual feedback and adjustment processes.

F. Planning

1) Reasoning Process: In the capabilities of foundation-model-based agent systems, the **reasoning process** is a crucial

component that utilizes cognitive steps and logical frameworks to tackle complex problems. The reasoning process bridges perception and action by enabling informed high-level decision-making and plan generation.

2) Tool Selection: In the planning process, selecting tools is essential for ensuring that agents have instant access to the necessary resources for efficient task execution. This list typically includes both internal and external tools, each playing a crucial role in the agent's architecture. **Internal tools** are tools and algorithms developed within the system to optimize performance and provide tailored solutions to operational challenges. Internal tools can include proprietary large language models (LLMs) and specific algorithms designed to handle unique tasks efficiently [44, 59]. Databases and knowledge bases form a robust backbone for agents, offering access to extensive repositories of structured data essential for tasks requiring comprehensive analytical capabilities [59, 73]. **External tools** extend the agent's capabilities beyond its inherent functions by allowing seamless interaction with external data sources and services. APIs facilitate this interaction, enabling the agent to access and utilize data from various external sources [59]. Additionally, external models are employed for specific tasks that demand specialized computational expertise. These models integrate cutting-edge algorithms to enhance the agent's problem-solving abilities and are typically used for more complex tasks involving multiple APIs [74]. By integrating both internal and external tools into the architecture, agents can achieve a higher level of flexibility and efficiency in task execution.

3) Plan Generation: To achieve the user's goal, the agent needs to translate the outcomes of the reasoning process into actionable steps. And accordingly, develop a plan [75]. This component is designed to operationalize the theoretical insights and strategies formulated during the reasoning phase into a coherent sequence of steps that guide the agent toward achieving specific objectives. There are two primary design options for plan generation: **single-path plan generator**, and **multi-path plan generator**.

Single-path plan generator creates a linear, straightforward plan that directs the agent from the start to the finish of a task without deviation. It follows a step-by-step methodology, ideal for tasks with predictable outcomes and clear procedures. The single-path approach ensures a focused and direct route to the goal, minimizing the possibility of errors and inefficiencies in execution [76]. **Multi-path plan generator** offering a more complex and adaptable planning solution, the multi-path plan generator devises several potential routes to achieve the goal [77, 78]. It allows the agent to dynamically adjust the steps in the plan based on new information or changes in the environment, thus enhancing its flexibility and effectiveness in uncertain scenarios.

G. Action

1) Agent Coordination: At the runtime stage, agent coordination mechanisms ensure that agents work together effectively, avoiding conflicts and redundancies. **Centralized coordination** involves a central agent or system managing the coordination of all agents [24, 79, 80]. This approach provides strong oversight but can become a bottleneck, particularly in large-scale systems. **Federated coordination** is managed in a decentralized manner, with each agent handling its own coordination while still communicating with others [8]. **Fully distributed coordination**, on the other hand, involves no central coordinator; all agents coordinate directly with each other, often using peer-to-peer communication protocols [24, 81, 82].

These coordination strategies are essential for ensuring that multi-agent systems operate efficiently and effectively. They help in managing dependencies, synchronizing actions, and resolving conflicts among agents. Each approach has its advantages and trade-offs in terms of efficiency, scalability, and complexity. Selecting the appropriate coordination mechanism depends on the specific requirements and constraints of the system.

2) Agent Communication: **Agent communication** strategies are crucial for the effective operation of multi-agent systems, as they dictate how agents share information and collaborate. There are several levels of communication transparency that can be implemented, each with its own advantages and trade-offs. **Full transparency:** All information is openly shared among agents. This maximizes cooperation and ensures that all agents have the same level of information, which can be particularly beneficial in collaborative environments. For instance, in a research setting where multiple agents are working together to solve complex problems, full transparency allows all agents to access and build upon each other's findings. However, full transparency can lead to information overload and may not be suitable for environments where privacy and security are critical concerns [83]. **Partial transparency:** `Goal-based sharing`: Information is shared only if it is relevant to achieving specific goals. For example, in a project management system, team members might only see tasks and data relevant to their specific objectives [24, 81]. `Role-based sharing`: Information is shared based on the roles and responsibilities of each agent [24, 84]. This is common in corporate environments where different agents have access to different levels of information. `Sensitive data withholding`: Sensitive information is withheld to protect privacy and security. This approach is crucial in scenarios where data sensitivity is high, and privacy regulations must be strictly followed [83]. `Context-aware sharing`: Information sharing is adapted based on the context and current state of the system. For instance, agents utilize intrinsic context such as their own historical data and goals, alongside extrinsic context including user preferences, other agents' behaviors, and systemic norms, to enhance decision-making [85].

Effective agent communication strategies are essential for maintaining the balance between collaboration and privacy. The choice of communication strategy depends on the specific requirements and constraints of the system, such as the need for real-time updates, security considerations, and the
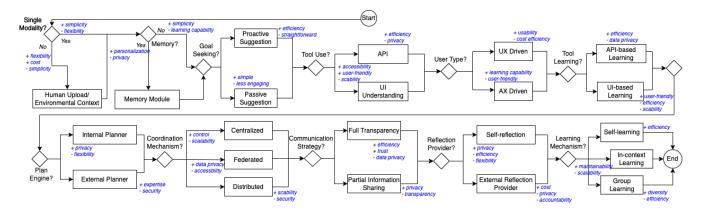
Fig. 3. Foundation Model-based Agent Decision Model

complexity of tasks being performed by the agents.

3) Tool Calling: Tool calling is a fundamental capability of foundation-model-based agents at the action stage, enabling them to access and manipulate a variety of external tools dynamically. **Tool invocation** involves methods that allow agents to interact with external tools either through `Configuration-free invocation` or `Customizable invocation` approaches. Configuration-free invocation enables agents to utilize tools via predefined interfaces without needing adjustments to the tools' configurations. This seamless and swift integration is ideal for routine operations, where standardization is crucial for efficiency. For instance, agents may use APIs for real-time data retrievals like weather or stock prices, leveraging methods described in tuning-free approaches to enhance parameter identification without manual tuning [86]. Customizable invocation involves agents dynamically adjusting the tools' parameters or modifying their processing strategies to optimize performance tailored to specific tasks [87]. This method is beneficial in environments requiring high precision and adaptability.

Output Integration: Within tool calling, explicitly incorporating outputs from external tools into the agent's decision-making process is crucial. `Inline integration` allows agents to directly use the outputs of tools in their workflows, ensuring quick and efficient task execution [59, 68]. Meanwhile, `multi-source integration` involves synthesizing outputs from multiple tools or combining these with the agent's internal data. This synthesis is critical for tasks requiring deep analysis and comprehensive responses, as it allows the agent to provide nuanced and context-aware outputs [58].

4) Actuation: **Actuation** [88] refers to the capability of an agent to take actions that affect the physical or digital environment. **Physical actuation** involves robotic movements and manipulations, such as a robot performing assembly tasks in a factory. **Virtual actuation** includes actions within software environments, such as automated data entry or network configuration changes. Effective actuation requires the development of sophisticated actuators and robust integration with control systems. For example, in autonomous vehicles,

actuation encompasses steering, braking, and acceleration, all coordinated by AI to ensure safe and efficient travel.

H. Reflection

1) Reflected Artifacts : In the reflection process of foundation-model-based agents, several key artifacts ensure effective traceability and observability. These artifacts can be categorized into three design options: **workflow/plan generation**, **intermediate result**, and **final output**. By providing feedback at different stages, we can ensure that not only the final output is correct, but also the intermediate processes. Workflow/plan generation involves the initial setup of goals, user inputs, prompts, and the overall planning steps that the agent will follow. Documenting these elements ensures that the agent has all the necessary information for decision-making and that its actions are aligned with intended objectives[89]. During this stage, feedback can be provided on the comprehensiveness and accuracy of the workflow, ensuring that the foundation for subsequent processes is solid.

The intermediate result includes the documentation of reasoning processes, planning steps, and tool use during the agent's operation. It involves logging intermediate decisions, actions, and outputs generated by the agent throughout its workflow. Feedback at this stage focuses on the correctness and rationality of these intermediate steps, ensuring that the process leading to the final output is valid and transparent [68]. The final output comprises the final results produced by the agent, which are directly evaluated against the goals and expectations set in the workflow/plan stage. Feedback here ensures that the output meets the desired criteria and accurately reflects the agent's intended actions and decisions.

2) Provider: Reflection allows the agent to incorporate feedback to refine the plan [50] from different providers, such as object world, the virtual environment, the interaction with humans, or the internal model. **Self-reflection** enables the agent to generate feedback on the plan and provides refinement guidance from themselves [44, 90, 91, 92, 93]. **Cross-reflection** uses different agents or FMs to provide feedback and refinement on the plan [93, 94, 95]. **Human reflection** means that the agent can collect feedback from

humans to refine the plan, which can effectively make the plan aligned with the human's preference [72, 96]. **Environmental reflection** allows agents to obtain feedback from the objective world or virtual environment, such as task completion signals or observations after an action. For example, ReAct uses thought-act-observation triplets for planning [90]. Voyager incorporates feedback from program execution and errors [44].

3) Mechanisms: Mechanisms refer to the specific processes and techniques employed by foundation-model-based agents to reflect on, assess, and adapt their behavior based on the artifacts they generate. These mechanisms are essential for the continuous improvement and accuracy of agent operations. The **Chain of Thought (CoT)** mechanism involves the agent articulating its reasoning process step-by-step, mirroring human cognitive processes. This method helps in tracing how decisions are made and identifying where errors might have occurred [77]. It is particularly useful in complex decision-making scenarios where justifications for each step are required for validation and learning purposes. **Tree of Thoughts (ToT)** extends the reflection process by allowing agents to explore multiple reasoning pathways simultaneously [78]. This mechanism helps agents to consider various possible outcomes before settling on a final decision, enhancing the robustness and depth of the decision-making process. It's akin to scenario analysis in human cognitive processes, where each branch of the tree represents a different line of reasoning or a different potential outcome.

I. Learning Capability

1) Training of Underlying Model: Effective training and learning mechanisms are essential for developing and adapting foundation-model-based agents. Various strategies can be employed, depending on the specific requirements and constraints of the system. **Centralized training** involves training underlying models using a centralized dataset and process, ensuring consistency but potentially requiring significant resources [97]. **Collaborative learning**, on the other hand, involves agents learning by collaborating. This includes `federated learning`, where each agent trains on local data and shares updates with a central model. For example, this work focuses on privacy-preserving federated training across multiple healthcare institutions, leveraging localized model training with global aggregation to enhance diagnosis without sharing sensitive data. [98]. `Distributed learning` is similar but emphasizes distributed computing resources more, as seen in environmental monitoring where distributed sensor networks share learning updates to enhance prediction accuracy. Another method is `split learning`, where the training process is divided between different agents or systems, sharing intermediate data.

`Online learning` allows agents to continue learning and adapting in real-time as they interact with their environment. A recommendation system that updates its model continuously based on user interactions and feedback can be an example of this approach [99]. These training mechanisms ensure that agents remain up-to-date and capable of handling new tasks, enhancing their adaptability and performance in dynamic environments.

2) Other Learning (without changing model weights): Learning capabilities are fundamental to an AI agent's ability to adapt and improve over time. **Self-learning** agents autonomously update their knowledge base and refine their decision-making algorithms [43]. A crucial aspect of this adaptation is the concept of long-term memory, where agents retain and utilize extensive historical data to enhance their learning and decision-making processes over extended periods [38]. This continuous learning process enables them to handle new situations more effectively and improve their performance without external input [31]. **In-context learning** enables agents to adapt to new tasks by interpreting input context, often through prompt engineering, without altering internal parameters. This streamlined approach is particularly effective for agents using pre-trained models, allowing them to learn and respond based on tailored prompts. **Group learning** involves multiple agents sharing knowledge and insights to enhance collective performance. This can take several forms, `peer-to-peer learning` allows agents to share knowledge directly with each other [100], and `hierarchical learning`, which involves different levels of agents sharing and processing information and tackling complex problems more efficiently [101].

Figure 3 illustrates a decision model that complements the taxonomy of foundation model-based agents by providing a structured approach to design decisions. This flow chart guides architects through key choices, balancing trade-offs in efficiency, flexibility, privacy, scalability, etc. It aids in navigating complex design processes by visualizing decision points and their impacts, thereby enhancing the practical application of the taxonomy in developing foundation model-based agents.

J. Non-Functional Aspects

1) Level of Autonomy: Autonomy levels in foundation-model-based agents vary significantly, determining how independently an agent can operate without human intervention. LangChain recently published a list of cognitive architecture patterns [102] that can reflect the autonomy level of agents. At the basic level, **single-function calls** allow agents to perform simple, predefined tasks. As complexity increases, agents can execute a **chain of function calls**, allowing for more complex behaviors and decision-making processes. Using **function models as routers**, agents can dynamically route tasks based on real-time data and conditions. **State machines** provide a structured way for agents to transition between different states based on specific triggers, enabling more sophisticated control over their actions. **Fully autonomous agents** represent the highest level of autonomy, capable of making independent decisions, learning from their environment, and adapting their behavior to new situations [103]. These agents can operate in dynamic and unpredictable environments, such as autonomous vehicles navigating through city traffic, leveraging advanced

sensors and machine learning algorithms to make real-time decisions.

2) Control of Underlying AI Technology: AI technology can be sourced in various ways, each offering different levels of autonomy and customization. These sources can be categorized into **sovereign AI**, **partially sourced AI**, and **fully externally sourced AI**.

- Sovereign AI: These are self-contained and fully autonomous systems developed and maintained entirely in-house. They do not rely on external assistance, ensuring complete control over the AI's capabilities and data [104].
- Partially Sourced AI: These systems combine internal development with external resources, allowing for self-customization or collaborative development. For instance, BloombergGPT [29] exemplifies a collaboratively developed AI, utilizing a unique hybrid of proprietary Bloomberg financial data and expansive public datasets to enhance domain-specific performance in financial applications.
- Fully Externally Sourced AI: These systems rely entirely on external AI models and services, eliminating the need for internal development [14].

3) RAI Aspects/Guardrails: Responsible AI (RAI) aspects and guardrails are critical considerations in the development and deployment of foundation-model-based agents. Our ongoing work includes a detailed taxonomy of guardrails that addresses various risks and ethical concerns associated with AI systems. This taxonomy provides a structured approach to implementing guardrails, ensuring that agents operate safely, ethically, and transparently. Additionally, we are working on a comprehensive AgentOps framework, which integrates these guardrails into the operational workflows of AI agents. This framework is detailed in a concurrent paper on AgentOps, offering methodologies for monitoring and controlling agent behavior. These efforts aim to enhance the reliability, accountability, and societal impact of foundation-model-based agents.

V. Threats to Validity

While our study presents a comprehensive taxonomy for foundation-model-based agents, it is important to acknowledge several potential threats to the validity of our findings.

Coverage Limitations: The taxonomy we developed may not encompass all possible variations and characteristics of AI agents or foundation model-based agents. Despite our systematic approach, the rapidly evolving nature of AI technology means new models and capabilities are continuously emerging. Our search included literature up until May 2024, and any developments beyond this period are not considered in our analysis. Consequently, some recent advancements and applications might have been missed.

Data Extraction and Synthesis: The process of data extraction and thematic coding, although rigorously validated. Different researchers might interpret and categorize information differently, which could influence the resulting taxonomy.

We mitigated this through collaborative validation involving multiple authors, but some degree of subjectivity remains.

Despite these limitations, we believe that the breadth and depth of our literature review and the robustness of our methodological approach provide a solid foundation for the proposed taxonomy. Our study included a substantial number of papers, ensuring a comprehensive overview that supports the validity of our conclusions. Future research should continue to refine and expand this taxonomy, incorporating new developments and addressing identified gaps.

VI. Conclusion

Foundation model-based agents are gaining increasing attention in various domains. However, researchers and developers face architectural challenges when designing these agents. Our previous work demonstrated a reference architecture to present an overview of agent design [40], while in this study, we provided a comprehensive taxonomy that categorizes foundation-model-based agents, addressing issues of standardization and enhancing our understanding of their various types and functionalities. By detailing how agents function across different roles and situations, this classification aids in designing more effective and adaptable agent system architectures, reducing clutter in the field and streamlining design and deployment processes. Our taxonomy also offers valuable insights into architectural options, enabling better integration and interoperability across different systems.

In our future work, we will explore how to apply this taxonomy in conjunction with existing architectural patterns. Additionally, we will further investigate architecture decisions related to foundation model-based agents, aiming to incorporate new technologies and applications to advance the field.

References

[1] L. C. Budler, L. Gosak, and G. Stiglic, "Review of artificial intelligence-based question-answering systems in healthcare," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 13, no. 2, p. e1487, 2023.

[2] D. K. Nguyen, G. Sermpinis, and C. Stasinakis, "Big data, artificial intelligence and machine learning: A transformative symbiosis in favour of financial technology," *European Financial Management*, vol. 29, no. 2, pp. 517–548, 2023.

[3] H. Chen, K. Yuan, Y. Huang, L. Guo, Y. Wang, and J. Chen, "Feedback is all you need: from chatgpt to autonomous driving," *Science China Information Sciences*, vol. 66, no. 6, pp. 1–3, 2023.

[4] T. Wang, P. Zheng, S. Li, and L. Wang, "Multimodal human–robot interaction for human-centric smart manufacturing: A survey," *Advanced Intelligent Systems*, vol. 6, no. 3, p. 2300359, 2024.

[5] T. Masterman, S. Besen, M. Sawtell, and A. Chao, "The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey," *arXiv preprint arXiv:2404.11584*, 2024.

[6] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[7] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2004.04906*, 2020.

[8] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "Autogen: Enabling next-gen llm applications via multi-agent conversation framework," *arXiv preprint arXiv:2308.08155*, 2023.

[9] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou *et al.*, "Metagpt: Meta programming for multi-agent collaborative framework," *arXiv preprint arXiv:2308.00352*, 2023.

[10] N. R. Mehta, N. Medvidovic, and S. Phadke, "Towards a taxonomy of software connectors," in *Proceedings of the 22nd international conference on Software engineering*, 2000, pp. 178–187.

[11] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen *et al.*, "Palm 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[12] S. K. Singh, S. Kumar, and P. S. Mehra, "Chat gpt & google bard ai: A review," in *2023 International Conference on IoT, Communication and Automation Technology (ICICAT)*. IEEE, 2023, pp. 1–6.

[13] M. S. Rahaman, M. Ahsan, N. Anjum, M. M. Rahman, and M. N. Rahman, "The ai race is on! google's bard and openai's chatgpt head to head: an opinion article," *Mizanur and Rahman, Md Nafizur, The AI Race is on*, 2023.

[14] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[15] M. GenAI, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[16] Taskade, "Top 11 open-source autonomous agents & frameworks: The future of self-running ai," 2024, accessed: 2024-05-28. [Online]. Available: https://www.taskade.com/blog/top-autonomous-agents/

[17] CrewAI, "Crewai," 2024, accessed: 2024-05-28. [Online]. Available: https://www.crewai.com/

[18] K. Pandya and M. Holia, "Automating customer service using langchain: Building custom open-source gpt chatbot for organizations," *arXiv preprint arXiv:2310.05421*, 2023.

[19] K. Mei, Z. Li, S. Xu, R. Ye, Y. Ge, and Y. Zhang, "Aios: Llm agent operating system," *arXiv e-prints, pp. arXiv–2403*, 2024.

[20] Z. Zhao, F. Zhao, Y. Zhao, Y. Zeng, and Y. Sun, "A brain-inspired theory of mind spiking neural network improves multi-agent cooperation and competition," *Patterns*, vol. 4, no. 8, 2023.

[21] S. Schwartz, A. Yaeli, and S. Shlomov, "Enhancing trust in llm-based ai automation agents: New considerations and future challenges," *arXiv preprint arXiv:2308.05391*, 2023.

[22] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin *et al.*, "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, no. 6, p. 186345, 2024.

[23] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff *et al.*, "The prompt report: A systematic survey of prompting techniques," *arXiv preprint arXiv:2406.06608*, 2024.

[24] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, "Large language model based multi-agents: A survey of progress and challenges," *arXiv preprint arXiv:2402.01680*, 2024.

[25] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou *et al.*, "The rise and potential of large language model based agents: A survey," *arXiv preprint arXiv:2309.07864*, 2023.

[26] X. Li, "A survey on llm-based agents: Common workflows and reusable llm-profiled components," *arXiv preprint arXiv:2406.05804*, 2024.

[27] Y. Xia, M. Shenoy, N. Jazdi, and M. Weyrich, "Towards autonomous system: flexible modular production system enhanced with large language model agents," *arXiv preprint arXiv:2304.14721*, 2023.

[28] Z. Wang, S. Mao, W. Wu, T. Ge, F. Wei, and H. Ji, "Unleashing cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration," *arXiv preprint arXiv:2307.05300*, 2023.

[29] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, "Bloomberggpt: A large language model for finance," *arXiv preprint arXiv:2303.17564*, 2023.

[30] G. Mialon, C. Fourrier, C. Swift, T. Wolf, Y. LeCun, and T. Scialom, "Gaia: a benchmark for general ai assistants," *arXiv preprint arXiv:2311.12983*, 2023.

[31] Z. Wu, C. Han, Z. Ding, Z. Weng, Z. Liu, S. Yao, T. Yu, and L. Kong, "Os-copilot: Towards generalist computer agents with self-improvement," *arXiv preprint arXiv:2402.07456*, 2024.

[32] B. A. I. R. (BAIR), "The shift from models to compound ai systems," *BAIR Blog*, February 2024, accessed: 2024-06-10. [Online]. Available: https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/

[33] L. Yi, H. Yu, C. Ren, H. Zhang, G. Wang, X. Liu, and X. Li, "Fedmoe: Data-level personalization with mixture of experts for model-heterogeneous personalized federated learning," *arXiv preprint arXiv:2402.01350*,

2024.

[34] K. Lu, H. Yuan, R. Lin, J. Lin, Z. Yuan, C. Zhou, and J. Zhou, "Routing to the expert: Efficient reward-guided ensemble of large language models," *arXiv preprint arXiv:2311.08692*, 2023.

[35] X. Wu, S.-h. Wu, J. Wu, L. Feng, and K. C. Tan, "Evolutionary computation in the era of large language model: Survey and roadmap," *arXiv preprint arXiv:2401.10034*, 2024.

[36] Y. Kang, T. Fan, H. Gu, L. Fan, and Q. Yang, "Grounding foundation models through federated transfer learning: A general framework," *arXiv preprint arXiv:2311.17431*, 2023.

[37] Y. Jin, X. Shen, H. Peng, X. Liu, J. Qin, J. Li, J. Xie, P. Gao, G. Zhou, and J. Gong, "Surrealdriver: Designing generative driver agent simulation framework in urban contexts based on large language model," *arXiv preprint arXiv:2309.13193*, 2023.

[38] A. Maharana, D.-H. Lee, S. Tulyakov, M. Bansal, F. Barbieri, and Y. Fang, "Evaluating very long-term conversational memory of llm agents," *arXiv preprint arXiv:2402.17753*, 2024.

[39] A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, and G. Huang, "Expel: Llm agents are experiential learners," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 19 632–19 642.

[40] Q. Lu, L. Zhu, X. Xu, Z. Xing, S. Harrer, and J. Whittle, "Towards responsible generative ai: A reference architecture for designing foundation model based agents," *arXiv preprint arXiv:2311.13148*, 2023.

[41] T. Shibata, G. Irie, D. Ikami, and Y. Mitsuzumi, "Learning with selective forgetting." in *IJCAI*, vol. 3, 2021, p. 4.

[42] Y. Fei, J. Li, and Y. Li, "Selective memory recursive least squares: Recast forgetting into memory in rbf neural network-based real-time learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[43] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[44] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.

[45] W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang, "Memorybank: Enhancing large language models with long-term memory," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 19 724–19 731.

[46] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun, "Communicative agents for software development," *arXiv preprint arXiv:2307.07924*, 2023.

[47] X. Zhou, G. Li, and Z. Liu, "Llm as dba," *arXiv preprint arXiv:2308.05481*, 2023.

[48] X. Zhu, Y. Chen, H. Tian, C. Tao, W. Su, C. Yang, G. Huang, B. Li, L. Lu, X. Wang *et al.*, "Ghost in the minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory," *arXiv preprint arXiv:2305.17144*, 2023.

[49] A. Modarressi, A. Imani, M. Fayyaz, and H. Schütze, "Ret-llm: Towards a general read-write memory for large language models," *arXiv preprint arXiv:2305.14322*, 2023.

[50] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023, pp. 1–22.

[51] H. Gao and Y. Zhang, "Memory sharing for large language model based agents," *arXiv preprint arXiv:2404.09982*, 2024.

[52] Z. Zhang, X. Bo, C. Ma, R. Li, X. Chen, Q. Dai, J. Zhu, Z. Dong, and J.-R. Wen, "A survey on the memory mechanism of large language model based agents," *arXiv preprint arXiv:2404.13501*, 2024.

[53] D. Zhao, Z. Xing, X. Xia, D. Ye, X. Xu, and L. Zhu, "Seehow: Workflow extraction from programming screencasts through action-aware video analytics," *arXiv preprint arXiv:2304.14042*, 2023.

[54] X. Zeng, X. Wang, T. Zhang, C. Yu, S. Zhao, and Y. Chen, "Gesturegpt: Zero-shot interactive gesture understanding and grounding with large language model agents," *arXiv preprint arXiv:2310.12821*, 2023.

[55] Y. Liu, S. Chen, H. Chen, M. Yu, X. Ran, A. Mo, Y. Tang, and Y. Huang, "How ai processing delays foster creativity: Exploring research question co-creation with an llm-based agent," *arXiv preprint arXiv:2310.06155*, 2023.

[56] S. S. Kannan, V. L. Venkatesh, and B.-C. Min, "Smartllm: Smart multi-agent robot task planning using large language models," *arXiv preprint arXiv:2309.10062*, 2023.

[57] K. Zhang, J. Li, G. Li, X. Shi, and Z. Jin, "Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges," *arXiv preprint arXiv:2401.07339*, 2024.

[58] Z. Durante, B. Sarkar, R. Gong, R. Taori, Y. Noda, P. Tang, E. Adeli, S. K. Lakshmikanth, K. Schulman, A. Milstein *et al.*, "An interactive agent foundation model," *arXiv preprint arXiv:2402.05929*, 2024.

[59] C. Qu, S. Dai, X. Wei, H. Cai, S. Wang, D. Yin, J. Xu, and J.-R. Wen, "Tool learning with large language models: A survey," *arXiv preprint arXiv:2405.17935*, 2024.

[60] G. Baechler, S. Sunkara, M. Wang, F. Zubach, H. Mansoor, V. Etter, V. Cărbune, J. Lin, J. Chen, and

A. Sharma, "Screenai: A vision-language model for ui and infographics understanding," *arXiv preprint arXiv:2402.04615*, 2024.

[61] K. You, H. Zhang, E. Schoop, F. Weers, A. Swearngin, J. Nichols, Y. Yang, and Z. Gan, "Ferret-ui: Grounded mobile ui understanding with multimodal llms," *arXiv preprint arXiv:2404.05719*, 2024.

[62] Å. Stige, E. D. Zamani, P. Mikalef, and Y. Zhu, "Artificial intelligence (ai) for user experience (ux) design: a systematic literature review and future research agenda," *Information Technology & People*, 2023.

[63] L. Brand, B. G. Humm, A. Krajewski, and A. Zender, "Towards improved user experience for artificial intelligence systems," in *International Conference on Engineering Applications of Neural Networks*. Springer, 2023, pp. 33–44.

[64] G. Research, "Ai in software engineering at google: Progress and the path ahead," 2024, accessed: 2024-06-28.

[65] X. Liu, X. Lou, J. Jiao, and J. Zhang, "Position: Foundation agents as the paradigm shift for decision making," *arXiv preprint arXiv:2405.17009*, 2024.

[66] S. Hao, Y. Gu, H. Ma, J. J. Hong, Z. Wang, D. Z. Wang, and Z. Hu, "Reasoning with language model is planning with world model," *arXiv preprint arXiv:2305.14992*, 2023.

[67] Universal Task Assistant, "Universal task assistant(uta)," https://apputa.online/, 2024, accessed: 2024-06-25.

[68] Y. Liu, S. K. Lo, Q. Lu, L. Zhu, D. Zhao, X. Xu, S. Harrer, and J. Whittle, "Agent design pattern catalogue: A collection of architectural patterns for foundation model-based agents," *arXiv preprint arXiv:2405.10467*, 2024.

[69] S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel, and D. Schuurmans, "Foundation models for decision making: Problems, methods, and opportunities," *arXiv preprint arXiv:2303.04129*, 2023.

[70] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "Llm+ p: Empowering large language models with optimal planning proficiency," *arXiv preprint arXiv:2304.11477*, 2023.

[71] Z. Liu, Y. Zhang, P. Li, Y. Liu, and D. Yang, "Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization," *arXiv preprint arXiv:2310.02170*, 2023.

[72] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[73] Y. Gu, Y. Shu, H. Yu, X. Liu, Y. Dong, J. Tang, J. Srinivasa, H. Latapie, and Y. Su, "Middleware for llms: Tools are instrumental for language agents in complex environments," *arXiv preprint arXiv:2402.14672*, 2024.

[74] Z. Yang, L. Li, J. Wang, K. Lin, E. Azarnasab, F. Ahmed, Z. Liu, C. Liu, M. Zeng, and L. Wang, "Mmreact: Prompting chatgpt for multimodal reasoning and action," *arXiv preprint arXiv:2303.11381*, 2023.

[75] Y. Ye, X. Cong, S. Tian, J. Cao, H. Wang, Y. Qin, Y. Lu, H. Yu, H. Wang, Y. Lin *et al.*, "Proagent: From robotic process automation to agentic process automation," *arXiv preprint arXiv:2311.10751*, 2023.

[76] Z. Wang, Z. Liu, Y. Zhang, A. Zhong, L. Fan, L. Wu, and Q. Wen, "Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models," *arXiv preprint arXiv:2310.16340*, 2023.

[77] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[78] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[79] S. Agashe, Y. Fan, and X. E. Wang, "Evaluating multi-agent coordination abilities in large language models," *arXiv preprint arXiv:2310.03903*, 2023.

[80] E. Pesce and G. Montana, "Learning multi-agent coordination through connectivity-driven communication," *Machine Learning*, vol. 112, no. 2, pp. 483–514, 2023.

[81] C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu, "Chateval: Towards better llm-based evaluators through multi-agent debate," *arXiv preprint arXiv:2308.07201*, 2023.

[82] F. Xu and T. Kaneko, "Local coordination in multi-agent reinforcement learning," in *2021 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE, 2021, pp. 149–154.

[83] C. Zhu, M. Dastani, and S. Wang, "A survey of multi-agent deep reinforcement learning with communication," *Autonomous Agents and Multi-Agent Systems*, vol. 38, no. 1, p. 4, 2024.

[84] A. E. Hassan, D. Lin, G. K. Rajbahadur, K. Gallaba, F. R. Cogo, B. Chen, H. Zhang, K. Thangarajah, G. A. Oliva, J. Lin *et al.*, "Rethinking software engineering in the foundation model era: A curated catalogue of challenges in the development of trustworthy fmware," *arXiv preprint arXiv:2402.15943*, 2024.

[85] H. Du, S. Thudumu, R. Vasa, and K. Mouzakis, "A survey on context-aware multi-agent systems: Techniques, challenges and future directions," *arXiv preprint arXiv:2402.01968*, 2024.

[86] C.-Y. Hsieh, S.-A. Chen, C.-L. Li, Y. Fujii, A. Ratner, C.-Y. Lee, R. Krishna, and T. Pfister, "Tool documentation enables zero-shot tool-usage with large language models," *arXiv preprint arXiv:2308.00675*, 2023.

[87] S. Qiao, H. Gui, C. Lv, Q. Jia, H. Chen, and N. Zhang, "Making language models better tool learners with

execution feedback," *arXiv preprint arXiv:2305.13068*, 2023.

[88] A. Chan, R. Salganik, A. Markelius, C. Pang, N. Rajkumar, D. Krasheninnikov, L. Langosco, Z. He, Y. Duan, M. Carroll *et al.*, "Harms from increasingly agentic algorithmic systems (2023)."

[89] L. Longo, M. Brcic, F. Cabitza, J. Choi, R. Confalonieri, J. Del Ser, R. Guidotti, Y. Hayashi, F. Herrera, A. Holzinger *et al.*, "Explainable artificial intelligence (xai) 2.0: A manifesto of open challenges and interdisciplinary research directions," *Information Fusion*, vol. 106, p. 102301, 2024.

[90] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[91] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang *et al.*, "Self-refine: Iterative refinement with self-feedback," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[92] T. R. Sumers, S. Yao, K. Narasimhan, and T. L. Griffiths, "Cognitive architectures for language agents," *arXiv preprint arXiv:2309.02427*, 2023.

[93] N. Shinn, B. Labash, and A. Gopinath, "Reflexion: an autonomous agent with dynamic memory and self-reflection," *arXiv preprint arXiv:2303.11366*, 2023.

[94] P.-L. Chen and C.-S. Chang, "Interact: Exploring the potentials of chatgpt as a cooperative agent," *arXiv preprint arXiv:2308.01552*, 2023.

[95] Y. Talebirad and A. Nadiri, "Multi-agent collaboration: Harnessing the power of intelligent llm agents," *arXiv preprint arXiv:2306.03314*, 2023.

[96] G. Sarch, Y. Wu, M. J. Tarr, and K. Fragkiadaki, "Open-ended instructable embodied agents with memory-augmented large language models," *arXiv preprint arXiv:2310.15127*, 2023.

[97] P. K. Sharma, R. Fernandez, E. Zaroukian, M. Dorothy, A. Basak, and D. E. Asher, "Survey of recent multi-agent reinforcement learning algorithms utilizing centralized training," in *Artificial intelligence and machine learning for multi-domain operations applications III*, vol. 11746. SPIE, 2021, pp. 665–676.

[98] A. Mabrouk, R. P. D. Redondo, M. Abd Elaziz, and M. Kayed, "Ensemble federated learning: An approach for collaborative pneumonia diagnosis," *Applied Soft Computing*, vol. 144, p. 110500, 2023.

[99] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[100] S. Peng, X. Hu, Q. Yi, R. Zhang, J. Guo, D. Huang, Z. Tian, R. Chen, Z. Du, Q. Guo *et al.*, "Self-driven grounding: Large language model agents with automatical language-aligned skill learning," *arXiv preprint arXiv:2309.01352*, 2023.

[101] A. Nash, A. Vardy, and D. Churchill, "Herd's eye view: Improving game ai agent learning with collaborative perception," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 19, no. 1, 2023, pp. 306–314.

[102] LangChain, "Openai's bet on a cognitive architecture," https://blog.langchain.dev/openais-bet-on-a-cognitive-architecture/, 2023, accessed: 2023-05-27.

[103] A. E. Hassan, G. A. Oliva, D. Lin, B. Chen, Z. Ming *et al.*, "Rethinking software engineering in the foundation model era: From task-driven ai copilots to goal-driven ai pair programmers," *arXiv preprint arXiv:2404.10225*, 2024.

[104] Y. LeCun, "A path towards autonomous machine intelligence. 2022," *URL https://openreview. net/p df.*