International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V74I2P104

Volume 74 Issue 2, 16-29, February 2026

Review Article

Agentic AI Systems: Architecture, Data Infrastructure, and Context Management

Hitesh Chugani

Independent Researcher, Charlotte, North Carolina, USA.

Corresponding Author : chugani.hitesh@gmail.com

Received: 12 December 2025 Revised: 17 January 2026 Accepted: 09 February 2026 Published: 26 February 2026

Abstract *- Artificial Intelligence has moved from reactive systems to systems that can reason, plan, and act over time. This shift has led to the development of Agentic Artificial Intelligence (Agentic AI). Unlike traditional AI systems that follow fixed input–output rules, agentic AI systems maintain internal state, reason over context, plan multi-step actions, and interact with external environments with limited human input. This review places agentic AI within the historical development of agent-based artificial intelligence. It describes the main components of agentic systems, including perception, memory, goal management, execution, reflection, and orchestration. It also explains the data infrastructure required to support these systems. Traditional enterprise data platforms and stateless API-based integrations are not designed to support long-term reasoning, continuous context use, or dynamic tool interaction, which limits their suitability for agentic AI. A central focus of the paper is the Model Context Protocol (MCP), which provides a stateful interoperability layer that connects agents with external tools and data sources. MCP is compared with BDI architectures, classical multi-agent systems, and REST-based integration to explain its role and limitations. Security, implementation, and governance issues are also discussed. The review identifies research gaps in coordination, benchmarking, data integration, and secure deployment. Overall, the paper shows that agentic AI builds on earlier agent-based concepts using neural methods and requires layered architectures that combine reasoning models with interoperability standards.*

Keywords *- Agentic AI, Model Context Protocol (MCP), Autonomous AI Agents, Interoperability, Autonomous Agents.*

1. Introduction

Artificial Intelligence (AI) has evolved from rule-based and reactive systems to systems that can reason, plan, and act independently (Figure 1). Traditional AI models were designed to perform predefined tasks or respond to specific inputs within fixed workflows. At the same time, recent developments focus on autonomy and goal-directed behaviour, which leads to the development of agentic Artificial Intelligence (AI). Agentic AI is defined as autonomous systems that can interpret goals, plan actions, coordinate tasks, and adapt to changing environments with minimal human intervention.

Agentic systems combine perception, reasoning, planning, and execution within a single framework. These systems often use Large Language Models (LLMs), reinforcement learning, and multimodal data to support multi-step problem solving over extended contexts.

As a result, agentic AI is applied in domains such as healthcare, finance, manufacturing, cybersecurity, logistics, and enterprise operations where traditional automation cannot handle dynamic conditions and complex decisions [1], [2], [3], [4]. This transition represents a shift from task-based automation to process-based management as intelligent agents manage workflows, track progress, handle errors, and improve outcomes with limited supervision [5], [6], [7].

Agentic AI systems are highly dependent upon effective context management. An agent must gather, maintain, and use contextual information from multiple sources to take appropriate actions.

This includes past interactions, structured databases, unstructured documents, and outputs from external tools. When an agent accesses and combines these diverse data sources, it produces consistent reasoning, structured plans, and adaptive responses.

However, most of the existing data infrastructures, such as data warehouses and data lakes, were developed for human analysis and batch processing. They are not optimized for the real-time, machine-oriented context exchange required by autonomous agents. With the expanding agentic systems, new data architectures and interoperability mechanisms are required to support autonomous operations [5], [7].

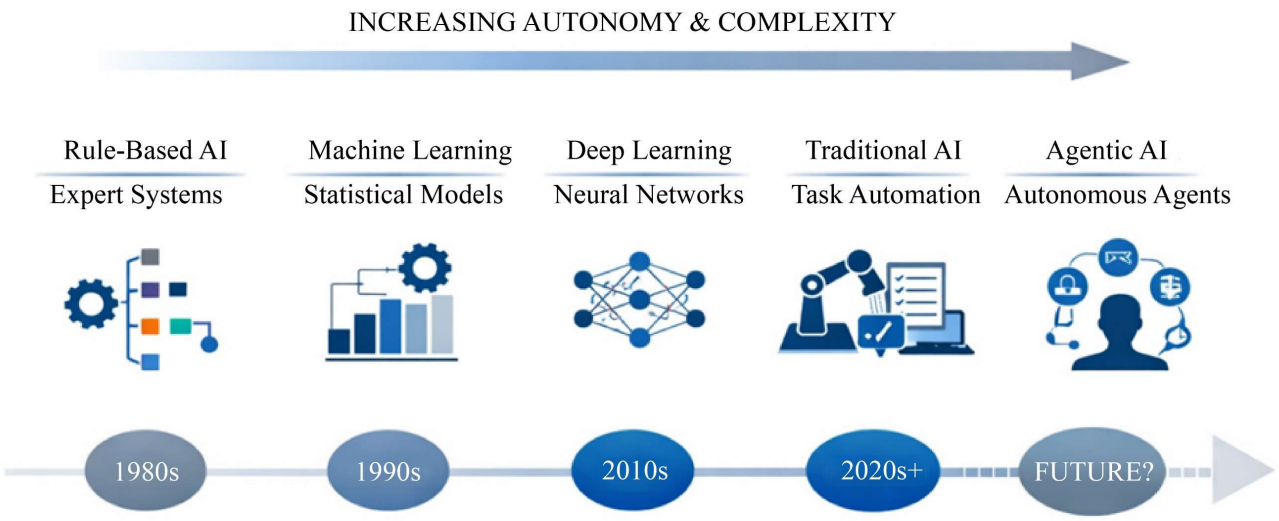

Fig. 1 Timeline of AI Evolution

Although agentic frameworks have advanced, the integration of multiple AI models, data repositories, and external software tools within a unified environment is difficult. The integration strategies are generally isolated and application-specific, which causes an M×N integration challenge. In this, each AI system has to build separate connections for every external resource it uses. Several frameworks, such as LangChain, LlamaIndex, CrewAI, AutoGen, LangGraph, Semantic Kernel, and MetaGPT, provide methods to link large language models with tools and data sources. However, these solutions operate within fragmented ecosystems, which reduces scalability and increases maintenance complexity.

These challenges indicate the need for standardized communication protocols with which agentic systems and external components can have consistent and scalable interaction [7], [8]. To address these limitations, the Model Context Protocol (MCP) was introduced in late 2024. It is an open standard for connecting AI systems with external tools and data sources. MCP defines a common interface with which agentic systems and large language models can access consistent contextual information. This reduces integration complexity and improves interoperability. MCP functions as a shared connection layer for AI applications and supports context-based and autonomous workflows [7], [9].

Recent review papers have studied agentic AI from different viewpoints. Ali et al. [5] have compared symbolic and neural approaches and discussed their theoretical and practical implications. Plaat et al. [10] introduced a reasoning–acting–interacting framework for LLM-based agents. Schneider [11] described the shift from generative AI to autonomous systems. Acharya et al. [12] examined methods that combine reinforcement learning with cognitive architectures, while Gridach et al. [13] explored the use of agentic systems in scientific discovery. Other works by Sapokta et al. [14] and Hosseini & Seilani [15] have focused on enterprise integration. Although these studies provide useful insights into architectures, reasoning models, and application domains, most of these concentrate on agent capabilities and conceptual autonomy. Less attention has been given to the infrastructure required for large-scale deployment, including context management, interoperability, integration, scalability, and communication standards. In particular, emerging standards such as the Model Context Protocol (MCP) have not been examined in detail within the broader evolution of agent-based AI systems. This review addresses these gaps by linking classical agent models with modern LLM-based architectures and by analysing the data and context layers needed for scalable agentic systems. It also examines interoperability challenges and evaluates MCP as a standard for supporting scalable agent ecosystems. By combining architectural, infrastructural, and protocol-level analysis, this study provides a system-level perspective on agentic AI.

2. Background

Agentic artificial intelligence has developed through a gradual progression of AI paradigms rather than a single breakthrough. To understand modern agentic systems, it is important to examine earlier approaches and distinguish between symbolic and neural models. During the Symbolic AI era (1950s–1980s), intelligent systems were based on logic and explicit knowledge representation. Expert systems showed that rule-based reasoning could achieve goal-oriented behavior, but in limited domains. However, these systems required manual rule creation and could not easily adapt to new situations, making they not scalable [16].

The Machine Learning era (1980s–2010s) shifted from rule-based programming to data-driven learning. AI systems learn patterns from data through statistical models, which are not dependent upon predefined rules. Reinforcement learning introduced the idea of agents that can interact with environments and improve actions based on rewards. It forms the perception–action loop, which is used in autonomous decision-making [17][18]. The Deep Learning era (2010s–present) further advanced AI through neural networks, which learn representations from large datasets. These models improved tasks such as image recognition and language processing, but most of them were task-specific systems rather than fully autonomous agents [19], [20].

The Generative AI era started with the development of transformer architectures, with which Large Language Models (LLMs) became capable of generating text, code, and other outputs across different tasks. These models can support reasoning and can be integrated into larger systems. On their own, LLMs are not autonomous agents, but they provide the core capability to develop more advanced autonomous systems [21], [22]. The Agentic AI era (2022–present) combines generative models with orchestration frameworks that include memory, planning, tool use, and repeated reasoning steps. Systems such as AutoGPT and multi-agent platforms show that agents can complete goal-based tasks across longer workflows. Symbolic agents follow fixed logical rules, but modern agentic systems rely on probabilistic generation supported by coordination mechanisms [5], [23]. This progression shows that modern agentic systems are not direct continuations of symbolic AI. Instead, they are based on neural and probabilistic models while still using core ideas such as goal orientation and perception–action loops. It is important to understand this difference for analyzing current system designs and addressing challenges related to interoperability, scalability, and infrastructure.

3. Research Methodology

This study uses a systematic literature review based on the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) framework. The primary aim is to examine research on agentic artificial intelligence, data infrastructure, interoperability frameworks, and the Model Context Protocol (MCP). Different databases like IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect, and Google Scholar were searched to collect relevant research data. Additionally, selected industry reports and technical documents from IBM were also reviewed.

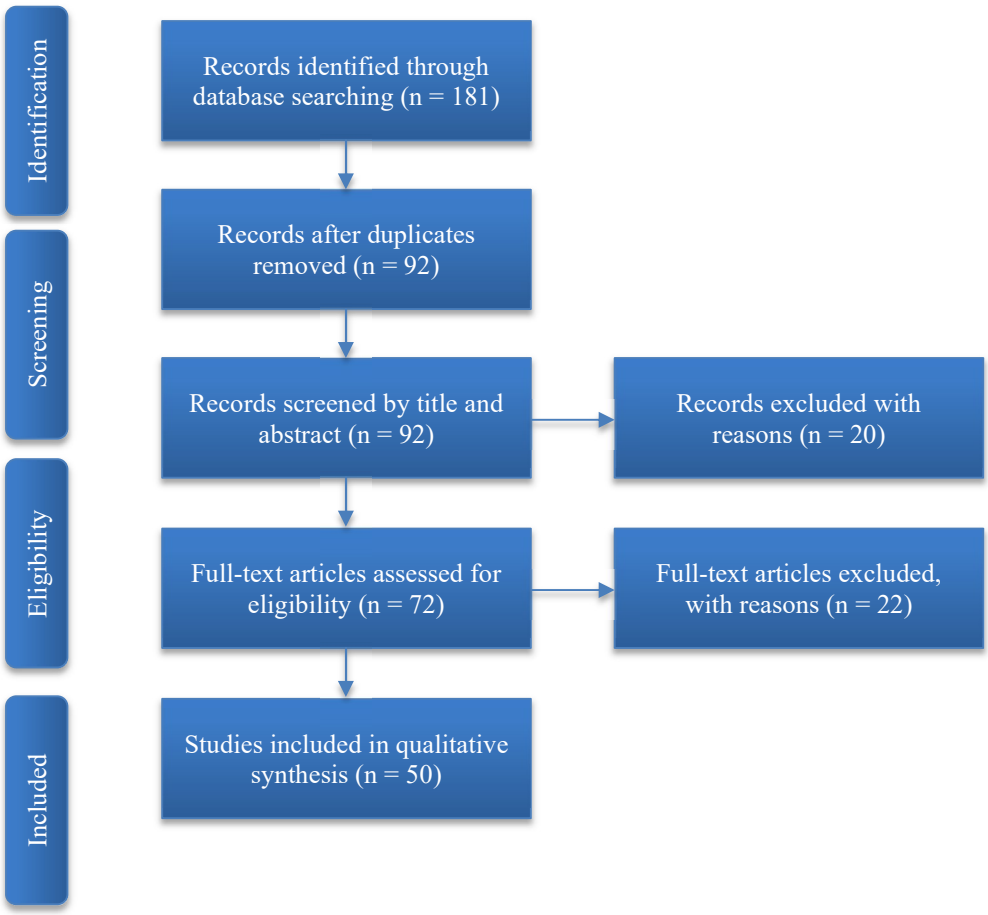

Fig. 2 PRISMA flow diagram for data extraction

Keywords like agentic AI, autonomous agents, multi-agent systems, AI architectures, context-aware systems, data infrastructure, interoperability, integration frameworks, orchestration frameworks, and MCP were used to search the databases. After collecting the research papers, first of all, duplicate records were removed. The titles and abstracts were screened for relevance, and after that, a full-text review was performed using predefined inclusion and exclusion criteria. Studies were included if they addressed agentic AI systems, autonomous agent architectures, data infrastructure or context management, integration or orchestration frameworks, or governance issues related to autonomous deployment. Studies were excluded if they focused only on traditional machine learning or deep learning models without agent frameworks, standalone large language models without orchestration, or narrow application areas without architectural relevance. After screening, about fifty sources were selected for analysis (Figure 2).

Data were extracted based on themes including the historical foundations of agent-based artificial intelligence, core architectural components of agentic AI, memory and goal management mechanisms, data infrastructure requirements, interoperability challenges, benchmarking against existing agent and integration paradigms, and the design and limitations of the Model Context Protocol (MCP). A qualitative synthesis approach was used to compare and organize findings. Despite all these, there must be a few limitations. Agentic AI and MCP are going through rapid changes, so there is a possibility that new developments may not yet appear in peer-reviewed research. In addition, industry reports may introduce bias, although this was reduced by selecting credible sources.

4. Results and Discussion

4.1. Summary of Reviewed Literature

Table 1 presents a structured summary of the key studies reviewed in this paper. The literature covers foundational theories of intelligent agents, the evolution from generative AI to agentic AI, architectural frameworks, multi-agent coordination, memory management, data infrastructure, and the Model Context Protocol (MCP). Several studies provide comprehensive surveys of agentic AI architectures and applications, while others focus on domain-specific implementations in healthcare, networking, scientific discovery, and industry. A growing body of work also examines interoperability through MCP, along with emerging discussions on security, governance, and protocol design. Together, these studies establish the conceptual, architectural, and operational foundations that inform current developments in agentic AI systems.

Table 1. Summary of Literature Review

Ref	Study / Authors (Year)	Type	Focus Area	Key Contribution
[1]	Toprani & Madisetti (2025)	Empirical (IEEE Access)	Agentic workflows	LLM-based agentic system for automated vulnerability detection and remediation in IaC
[2]	Chatzistefanidis et al. (2024)	Applied Research	Multi-agent automation (6G networks)	LLM-driven collaborative automation in intent-based networks
[3]	Karunanayake (2025)	Conceptual	Healthcare transformation	Explores agentic AI in clinical decision-making
[4]	Choi & Yoo (2026)	Perspective	MCP + Multimodal AI	Discusses the future of agentic systems in clinical medicine
[5]	Ali et al. (2026)	Comprehensive Survey	Architectures & Applications	Broad survey of agentic AI architectures and future directions
[10]	Plaat et al. (2025)	Survey (JAIR)	Agentic LLMs	Systematic overview of agentic LLM capabilities and planning
[11]	Schneider (2025)	Conceptual Survey	Generative → Agentic transition	Framework differentiating generative and agentic AI
[12]	Acharya et al. (2025)	Survey (IEEE Access)	Autonomous intelligence	Reviews goal-directed AI systems and architectures
[13]	Gridach et al. (2025)	Survey	Scientific discovery	Agentic AI in research automation
[14]	Sapkota et al. (2026)	Taxonomy Study	AI Agents vs Agentic AI	Clear conceptual differentiation and taxonomy
[15]	Hosseini & Seilani (2025)	Systematic Review	Smart systems	Reviews agentic AI for smart environments

[23]	Piccialli et al. (2025)	Survey	Industry 4.0	Autonomous agents in distributed industrial systems
[33]	Bandi et al. (2025)	Review	Definitions & Metrics	Comprehensive review of frameworks and evaluation metrics
[7]	Ayyagari (2025)	Technical Study	MCP interoperability	Explains MCP for contextual interoperability
[42]	Hou et al. (2025)	Landscape Analysis	MCP security	Reviews MCP security threats and risks
[48]	Ntousakis et al. (2025)	Conference Paper	MCP security & governance	Risk assessment and governance framework for MCP
[49]	Kumar et al. (2025)	Applied Security	MCP GUARDIAN	Security layer for safeguarding MCP-based systems
[46]	Rath (2025)	Conceptual	Adaptive MCP	Proposes trust-aware MCP extensions
[26]	Frering et al. (2025)	Applied Research	BDI + LLM integration	Combines Belief–Desire–Intention with LLMs for explainability
[24]	Brännström et al. (2022)	Theoretical	Bounded rational agents	Formal framework for rational agents
[31]	Liang et al. (2025)	Survey	Multi-agent reinforcement learning	Reviews MARL algorithms and coordination
[38]	Yu et al. (2025)	Technical Paper	Agentic memory	Unified long-term and short-term memory management
[40]	Varthakavi (2025)	Industry Article	Data architecture	Data-first architecture for agentic AI
[44]	Venkiteela (2025)	Conceptual	MCP vs APIs	Positions MCP as an interoperability paradigm

4.2. Historical Foundations of Agent-Based Artificial Intelligence

The idea of autonomous agents in artificial intelligence existed before the rise of large language model (LLM)-based systems. Current agentic AI is based on earlier research in rational agents, planning systems, multi-agent systems, and reinforcement learning. A clear understanding of these earlier approaches helps to place modern agentic architectures within the broader development of agent-based AI.

4.2.1. Rational Agent Model

The rational agent model is a standard framework for describing intelligent behaviour in AI. In this model, an agent perceives its environment through sensors and acts through actuators to achieve specific goals. Rationality is defined in terms of a performance measure: the agent chooses actions that maximize expected goal achievement based on its percept history and available knowledge.

This framework established the perception–reasoning–action loop as the core structure of intelligent systems. Within this model, agents are categorized as simple reflex agents, model-based agents, goal-based agents, and utility-based agents. These classifications formalized key concepts such as autonomy, goal orientation, and decision-making under uncertainty, which continue to influence modern agentic systems [24], [25].

4.2.2. Belief–Desire–Intention (BDI) Architecture

Alongside rational agent models, the Belief–Desire–Intention (BDI) architecture was developed as a framework for modeling deliberative agents. Based on theories of practical reasoning, BDI systems describe an agent in terms of its beliefs (information about the world), desires (goals), and intentions (committed plans). During the 1990s and early 2000s, BDI architectures were widely used in agent-oriented programming and distributed systems. The model defined concepts such as long-term planning, intention revision, and goal persistence. These ideas are similar to modern agent workflows that involve task decomposition, memory use, and plan adjustment. Although traditional BDI systems were based on symbolic logic and explicit rules, their structural ideas continue to affect the current reasoning frameworks of agents [26], [27].

4.2.3. Automated Planning Systems

Automated planning research also supported the development of agent autonomy. Early planning systems, such as STRIPS, gave formal ways to represent actions, preconditions, and effects. Classical planners generated sequences of actions that moved a system from an initial state to a goal state [28]. Later, Hierarchical Task Networks (HTN) allowed tasks to be divided into smaller sub-tasks. These methods established goal-directed behavior and structured

action sequencing [29]. Similar ideas are used in modern agent frameworks that perform multi-step reasoning and repeated execution. Although classical planners worked in symbolic state spaces. They provided the basic structure for planning, which modern LLM-based agents now approximate by using probabilistic methods.

4.2.4. Multi-Agent Systems (MAS)

Research in Multi-Agent Systems (MAS) expanded the focus from single agents to environments with multiple autonomous agents that interact through cooperation, negotiation, or competition. In the late 1990s and early 2000s, MAS studies examined coordination methods, distributed problem solving, and communication languages such as KQML and FIPA-ACL. These languages defined structured message formats and interaction rules for agents to exchange information and coordinate actions [30]. Many ideas from MAS research are reflected in current multi-agent systems, which are built on large language models. Modern collaborative and role-based agent frameworks use structured communication, role assignment, and distributed reasoning. These systems can be understood as updated implementations of earlier distributed AI concepts that are now supported by neural models instead of symbolic methods.

4.2.5. Reinforcement Learning Agents

Reinforcement Learning (RL) is developed alongside symbolic agent research. It introduced a mathematical approach to decision-making based on reward optimization. Using Markov Decision Processes (MDPs), RL agents learn policies by interacting with an environment and receiving feedback. Early RL methods used simple tabular representations. In the 2010s, the combination of RL with deep neural networks led to deep reinforcement learning, which allowed agents to operate in high-dimensional state spaces. Systems such as Deep Q-Networks (DQN) and AlphaGo showed that agents could learn complex strategies directly from raw inputs. RL established a closed interaction cycle: act, observe, evaluate, and update [31], [32]. This is similar to the iterative reasoning loops that are used in modern agent systems. Later methods, such as Reinforcement Learning from Human Feedback (RLHF), introduced human feedback into the training process to align agent behavior with human preferences.

4.2.6. Transition from Symbolic to Neural Agency

Early agent architectures were mainly based on symbolic rules, but they introduced core ideas such as goal representation, state modelling, planning, coordination, and feedback-based adaptation. Modern agentic AI systems replace symbolic reasoning with neural and probabilistic methods, but they keep these basic structures. Large language models act as reasoning components, while memory systems, planning modules, and tool-use mechanisms provide functions similar to those defined in rational agents, BDI models, and classical planning systems. Therefore, modern agentic AI should be seen as a continuation of earlier agent research rather than a complete break from it. It represents a neural implementation of established agent-based ideas [5]. Understanding this continuity helps explain the strengths and limits of current architectures and supports analysis of interoperability, context management, and standards such as the Model Context Protocol (MCP).

4.3. Agentic AI: Core Components and Architecture

Agentic Artificial Intelligence (Agentic AI) refers to AI systems that operate autonomously by setting goals, reasoning over context, planning actions, and interacting with external environments (Figure 3). Traditional AI systems mainly respond to inputs by using predefined mappings, but agentic AI systems maintain internal state, perform multi-step reasoning, and adapt to changing conditions. With this, they are able to complete goal-based tasks with limited human involvement, which makes them suitable for dynamic environments (Figure 4) [5], [6], [12], [33].

Organizations are now looking for automation that can perform practical tasks rather than only generate content, which is why interest in agentic AI is increasing. Industry reports estimate that the AI agent market was at about USD 5.3–5.4 billion in 2024, which is projected to be around USD 50–52 billion by 2030 with an annual growth of about 41–46% [34], [35]. At the architectural level, agentic AI systems consist of modular components that operate within a perception–decision–action loop. System behavior results from the interaction of these components rather than from a single model (Figure 5).

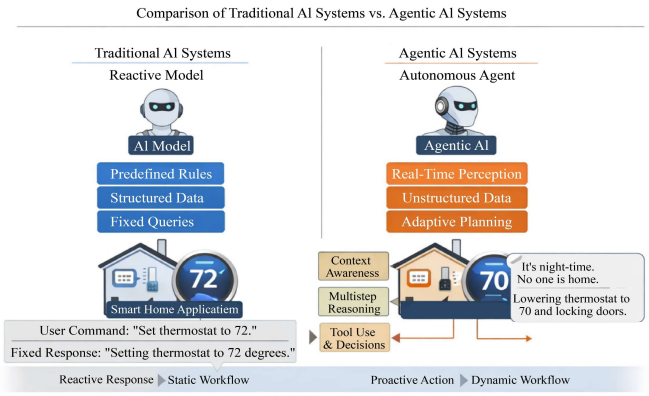

Fig. 3 Comparison of traditional AI systems and agentic AI systems

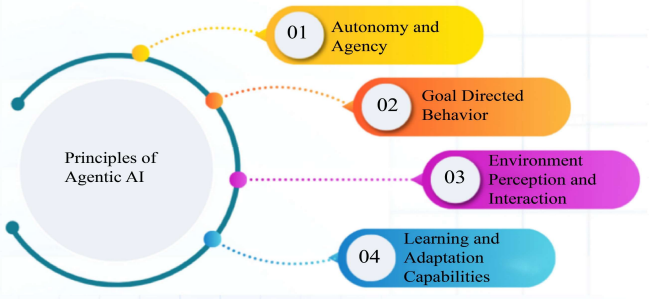

Fig. 4 Principles of agentic AI

4.3.1. Perception and World Representation

The perception component processes inputs such as text, sensor data, system events, or API responses and converts them into structured representations for reasoning. A state model maintains internal information about the environment, including observed and inferred data [5], [6], [33], [36], [37].

4.3.2. Memory Mechanisms

Memory mechanisms support continuity across tasks. Short-term memory stores current context, goals, and reasoning Fig 5.

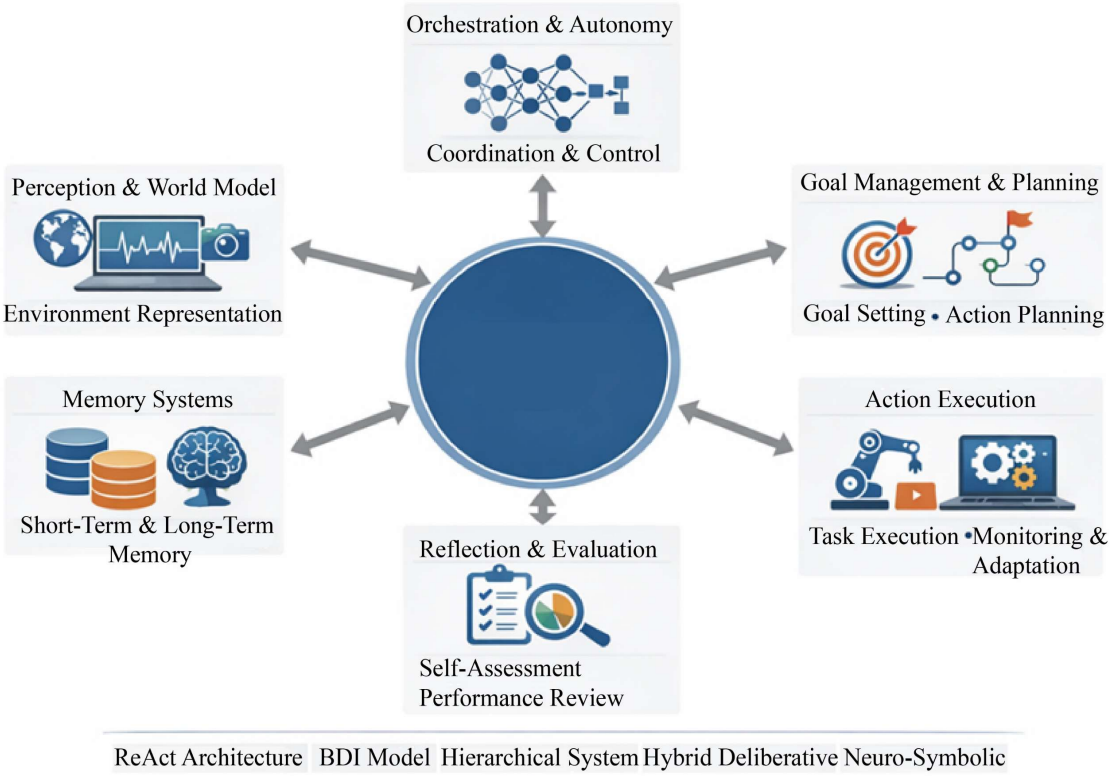

Fig. 5 Agentic AI: Core components and architecture

Steps, and recent interactions. Long-term memory stores past experiences, patterns, and historical data. Retrieved memory can influence current decisions and support adaptation [6], [33], [36], [37], [38], [39].

4.3.3. Goal Management, Planning, and Reasoning

Agentic AI systems operate using goal management, planning, and reasoning. High-level goals are divided into smaller sub-goals. Planning mechanisms evaluate options, organize actions, and select next steps. These processes may use symbolic, probabilistic, or learning-based methods and can support both short-term and long-term tasks [5], [6], [33], [36], [37].

4.3.4. Action Execution and Environment Interaction

The execution component carries out decisions by calling tools, APIs, workflows, or physical devices. It monitors results and compares them with expected outcomes. If differences occur, then the system can adjust its plans. This supports operation in changing environments [5], [6], [33], [36], [37].

4.3.5. Reflection and Evaluation

Reflection mechanisms allow systems to review their reasoning and outcomes. This may include checking plans, identifying errors, and adjusting future actions. Reflection can improve performance but may increase computational cost [5], [6], [33].

4.3.6. Orchestration and Autonomy

Orchestration components coordinate perception, memory, planning, execution, and reflection. In many systems, a large language model manages task sequencing, subtask delegation, and tool use. Autonomy results from the continuous interaction of these components [5], [6], [33].

4.3.7. Architectural Models

Agentic AI architectures can follow different models. These include ReAct-style architectures that combine reasoning and action, hierarchical systems with supervisory agents, Belief–Desire–Intention (BDI) models, reactive–deliberative systems, and neuro-symbolic approaches that combine symbolic and neural methods. Each model offers

different trade-offs in flexibility, interpretability, and control [5], [33].

4.4. Data Requirements for Agentic AI Systems

Agentic AI systems require a data infrastructure that differs from traditional enterprise analytics systems. Existing data architectures must be reviewed and redesigned to support their operation as autonomous agents are adopted in enterprise environments. Conventional enterprise data systems are designed for structured data, fixed queries, and predictable workflows. They support rule-based automation and static analysis but do not support probabilistic reasoning, semantic processing, or contextual inference required by agentic AI. Running agentic systems on legacy platforms often leads to systems that are difficult to scale and unreliable for real-time use. Agentic AI also depends heavily on unstructured data, such as documents, emails, conversations, images, and system logs. These sources contain important context but are not easily processed by traditional relational systems. Therefore, data infrastructures must support the ingestion, organization, and retrieval of unstructured data while preserving meaning and relationships [40], [41].

Agentic systems also require reliable access to large contextual inputs. They depend on complete, current, and consistent data because they perform multi-step reasoning over extended tasks. Incomplete or outdated information can reduce decision quality. Data systems must therefore reduce dependency risks and support stable access in distributed environments. Low latency and scalability are also important. Agents often interact with tools and external systems in real time. If data retrieval is slow or cannot support concurrent access and semantic search, system performance declines. Scalable infrastructure is needed to handle varying workloads and maintain consistent performance [40], [41].

The shift toward agentic AI reflects a move from rule-based data processing to language-based systems. As agent autonomy increases, data quality, diversity, and context become central to system performance. Data infrastructures must support semantic indexing, classification, and context-based retrieval instead of only storage and keyword search. Systems that combine general language models with domain-specific data also rely on these capabilities [40], [41].

4.5 Model Context Protocol (MCP) for Agentic AI Systems

The rapid growth of agentic AI systems has increased the need for scalable, context-aware methods that enable autonomous agents to interact with external tools, data sources, and services. Agentic systems operate across extended workflows, which involve multiple decision steps and tool calls. Their performance depends on maintaining context across different environments [7]. In 2024, the Model Context Protocol (MCP) was introduced to address these issues through a stateful, context-focused communication model designed for AI workflows [42]. MCP provides persistent context across interactions and standardizes how agents access data, execute actions, and reuse interaction patterns through resources, tools, and prompts. It does not replace existing APIs; instead, MCP functions as an abstraction layer above them. It reduces integration complexity and remains compatible with current systems [4], [7], [9], [43], [44].

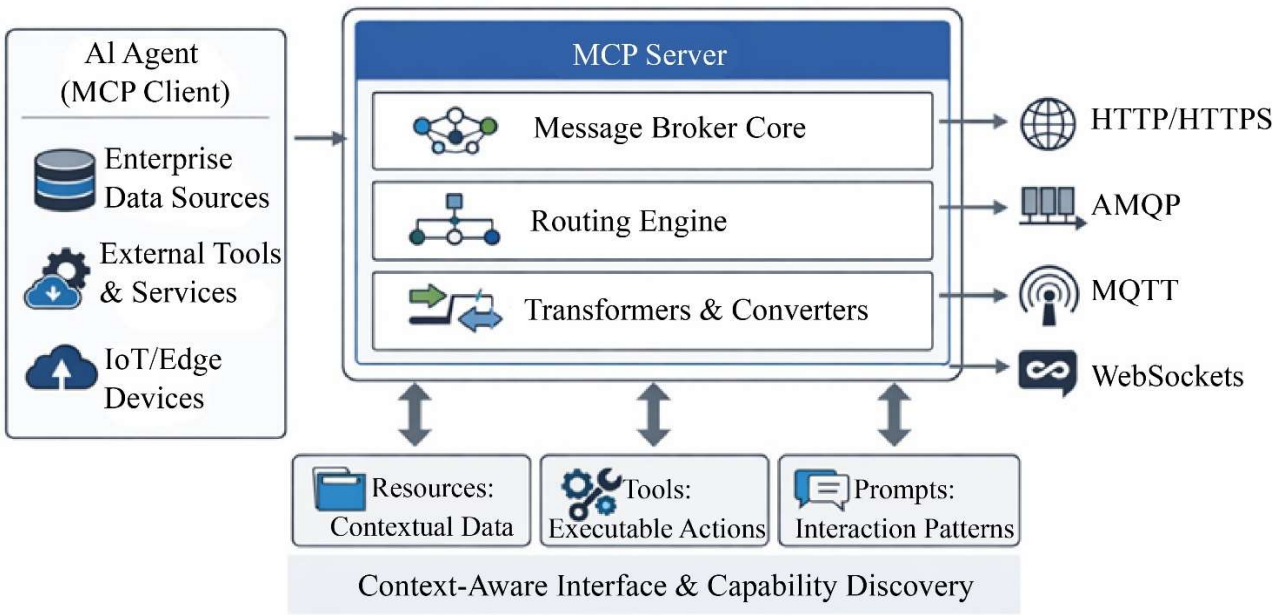

Fig. 6 Main components of MCP

4.5.1. Motivation and Background

Traditional integration methods use custom connectors to link AI models with specific tools or data sources. As the number of agents and services increases, this creates an "M×N" integration problem. Each new connection adds more maintenance work, fragments system design, and reduces reliability. Messaging protocols such as AMQP and MQTT improve data transfer and reliability in enterprise and IoT systems. However, they often require domain-specific adjustments and do not provide built-in support for maintaining semantic context across interactions. Enterprise integration frameworks, such as Enterprise Service Buses (ESBs), were developed to simplify system connectivity. However, they can introduce latency and additional architectural layers, which makes them less suitable for adaptive and time-sensitive AI systems [4], [7].

4.5.2. MCP Architecture and Core Components

MCP follows a client–server architecture with three main components: the MCP Host, MCP Client, and MCP Server (Figure 6). The MCP Host is the main AI application, such as an AI assistant or development tool, and manages communication between the AI model and external systems. Inside the host, the MCP Client creates and maintains a stateful connection with an MCP Server. The MCP Server runs as a separate service and provides external functions through a standard MCP interface. These functions include access to data, execution of actions, and reuse of interaction workflows. By separating AI applications from direct tool or service connections, MCP supports scalable integration across environments [7], [9], [43].

Within the MCP Server, there are three main components. A message handling core manages message exchange. A routing engine directs messages based on defined rules and context. Transformation modules handle schema mapping, data filtering, and format conversion to support interoperability. This structure allows MCP to work with communication protocols such as HTTP/HTTPS, AMQP, MQTT, and WebSockets. It can also be extended using plugin-based protocol extensions [7], [9], [43].

4.5.3. Communication Model and Interaction Primitives

Communication between MCP Clients and MCP Servers uses JSON-RPC 2.0. It supports transport methods such as standard input/output, HTTP, and WebSockets. The communication is stateful, which allows context to continue across multiple interactions. This is important for multi-step reasoning and autonomous decision-making in agentic systems. MCP defines a set of standard elements to structure these interactions. Resources provide access to contextual data, such as files, database records, or outputs from external services. Tools represent executable functions that agents call to perform actions or start workflows. Prompts define reusable interaction patterns that guide agent behavior in specific tasks. Together, these elements create a shared interaction model that reduces integration complexity and supports consistent behavior across workflows. MCP also includes features such as runtime capability discovery, server notifications, context scoping, and health checks. Capability discovery allows agents to query MCP Servers at runtime to identify available resources, tools, and prompts, which supports adaptive behavior without fixed integration logic [7], [9], [45].

4.5.4. MCP Enabling Contextual Interoperability for Agentic AI

A main contribution of MCP is its support for interoperability across different systems while maintaining context. It works as an abstraction layer that exposes underlying interfaces through a common protocol designed for AI systems. This reduces integration complexity, since each system needs to implement MCP only once to connect within an agent ecosystem. Context interoperability is important for agentic AI. Autonomous agents must maintain context not only during a single interaction but also across multiple actions, tool calls, and decision steps. MCP's stateful communication and standard interaction elements support this continuity. This allows agents to use previous states, adjust goals, and coordinate actions over time. In multi-agent environments, shared context also supports coordination and joint decision-making. By standardizing how context, data, and actions are exchanged, MCP enables agents to operate across extended workflows in areas such as enterprise systems, healthcare, finance, and IoT [7], [46].

4.6. Benchmarking MCP Against Agent Architectures and Integration Paradigms

A proper evaluation of MCP requires a clear definition of its role in agentic AI systems. MCP operates at the interoperability and infrastructure layer. It standardizes how agents access contextual data, discover available functions, and call external tools. In contrast, cognitive models such as Belief–Desire–Intention (BDI) define internal reasoning processes. Classical Multi-Agent Systems (MAS) focus on coordination among distributed agents, while REST and API-based methods provide service connectivity. These approaches operate at different levels. The following comparison does not treat them as alternatives but places them in a layered system view to explain their roles, strengths, and limits.

4.6.1. Comparison with Belief–Desire–Intention (BDI) Architectures

The BDI model is a cognitive architecture that describes how agents manage beliefs, goals, and intentions over time. It defines how agents reason, revise intentions, and maintain goals. Its main focus is internal reasoning rather than external integration [31], [32]. MCP differs in scope. It does not define belief structures or goal management. Instead, it standardizes the way agents exchange context and invoke tools [7]. BDI supports structured reasoning, while MCP supports system

integration and context management. In a layered view, BDI operates at the cognitive layer, and MCP operates at the interoperability layer. These approaches can work together. BDI can guide reasoning, and MCP can support tool coordination and context sharing across systems.

4.6.2. Comparison with Classical Multi-Agent Systems (MAS)

Classical Multi-Agent Systems (MAS) introduced communication standards such as FIPA-ACL and KQML to support structured interaction between agents. These systems define message types and coordination rules. MAS frameworks focus on distributed reasoning, negotiation, and group behavior. Their main strength is defining how agents coordinate and exchange messages in distributed environments. However, MAS protocols were designed for structured agent systems and do not directly address enterprise integration, cloud services, or diverse APIs. They also do not include built-in support for context persistence or runtime capability discovery, which must be handled separately [30]. MCP differs in focus. It does not standardize communication between agents. Instead, it standardizes how agents access tools and contextual data. Its stateful sessions and runtime capability discovery support integration across systems. MAS provides support for multi-agent coordination, while MCP supports interoperability at the infrastructure level.

4.6.3. Comparison with REST/API-Based Integration

RESTful APIs and RPC-based models are commonly used to connect systems in distributed environments. They are widely adopted and simple to use. However, they are usually stateless and depend on fixed endpoints and schemas. As the number of agents and services increases, REST-based integration can lead to the M×N integration problem, where each new connection adds complexity. MCP addresses these limits by introducing standard interaction elements, stateful context management, and runtime capability discovery. Instead of building separate connectors for each service, agents communicate through a common protocol layer. This reduces fragmentation and allows context to continue across multi-step workflows. However, MCP also has trade-offs. Stateful communication increases system complexity and requires session management and scalability planning. REST-based systems remain suitable for simple, stateless interactions with limited coordination needs [7], [9], [47].

Table 2. Comparative Evaluation of MCP, BDI Architecture, Classical Multi-Agent Systems, and REST/API Integration Frameworks

Evaluation Dimension	MCP	BDI Architecture	Classical MAS	REST/API Integration
Primary Focus	Interoperability and context-layer standardization	Cognitive reasoning and goal deliberation	Agent-to-agent coordination and messaging	Service connectivity
Reasoning Model	Not defined (model-agnostic)	Formal belief–desire–intention semantics	Distributed interaction semantics	None
Context Persistence	Stateful session-based	Internal mental state	Message-level unless extended	Stateless
Tool Invocation	Standardized and dynamic	External connectors required	Not standardized	Endpoint-based
Capability Discovery	Runtime discovery supported	Not native	Limited	Not supported
Multi-Agent Coordination	Partial (infrastructure-level)	Possible but not intrinsic	Strong	Weak
Scalability in Heterogeneous Environments	High	Moderate (integration-dependent)	Moderate	Low (M×N problem)
Implementation Complexity	Moderate to High	Moderate	High	Low
Maturity Level	Emerging	Mature	Mature	Highly mature

4.7. Application of MCP for Agentic AI

The Model Context Protocol (MCP) works as a middleware layer between AI systems and external tools. It standardizes how agents access context and execute functions. Instead of handling different API formats, agents use a common interface. This reduces development effort and removes the need for custom connectors. MCP also supports multi-agent systems by allowing agents to share context and access common tools without direct connections between each pair of agents. It can support retrieval-augmented generation (RAG) by treating retrieval as a tool call. Agents can connect to databases or knowledge stores through MCP servers and use retrieved data in multi-step tasks. At the system level, MCP separates AI applications from specific service providers, which reduces integration work. It supports structured data access and runtime capability discovery, with which agents identify and use available tools [7].

MCP can be applied in several domains. In software development, it provides access to code repositories and

documentation. In customer support, agents use customer records and product data. In data analytics, MCP maintains context across querying and reporting steps. Enterprise search systems can access documents and databases through one interface. In finance, MCP supports market analysis and connections to trading systems. In manufacturing, it provides access to machine data. In healthcare, it supports access to electronic health records for decision support [7]. MCP also supports workflows across multiple systems, where agents coordinate actions through a single interface. This applies to robotics, personal assistants, and enterprise systems [7].

4.8. Security and Implementation Considerations for MCP-Based Agentic Systems

While the Model Context Protocol (MCP) improves interoperability, its deployment introduces security and architectural trade-offs. Because MCP supports dynamic capability discovery, tool invocation, and stateful context exchange, it expands the system surface compared to stateless APIs. Secure implementation requires access control, protected context storage, and proper session management. One security issue is dynamic tool invocation. Runtime capability discovery increases flexibility but may allow unauthorized or unintended execution. Capability scoping, role-based access control, and authenticated client–server communication are needed to prevent misuse. Tools should operate under defined policies that limit their usability in all conditions. Context integrity is also important. Since MCP maintains session state, the stored context can become a target for attacks. Encryption, secure transport, input validation, and audit logging help to reduce risks such as context manipulation or injection. In multi-agent systems, session isolation is needed to prevent data leakage between agents [48], [49], [50].

From an implementation view, MCP's stateful design increases system complexity as compared to stateless REST systems. Session management and context synchronization must be handled carefully. Distributed systems must address replication, fault tolerance, and latency to maintain performance. Logging and monitoring are required to support reliable operation. In regulated domains, MCP systems should also support traceability of context changes and tool calls to enable accountability and compliance [48], [49], [50].

4.9. Critical Assessment of MCP

If MCP is critically assessed, then it has several limitations. First, MCP operates at the infrastructure layer and does not define how agents reason. It standardizes access to tools and context but does not manage beliefs, goals, intention updates, or decision guarantees [7], [42], [44]. As a result, MCP can support integration without ensuring consistent reasoning or long-term planning. Second, MCP has limited support for coordination in multi-agent systems. It allows agents to share tools and context but does not define mechanisms for negotiation, consensus, conflict resolution, or commitment management. Classical multi-agent system frameworks provide stronger support in these areas [42], [44]. Therefore, MCP alone may not be sufficient for systems that require structured coordination.

Third, dynamic capability discovery and stateful sessions introduce security and governance concerns. Runtime tool invocation increases the risk of unauthorized access or misuse. These risks can be reduced through external access control and policy mechanisms, but MCP does not include a complete security model at the protocol level [42], [48]. Fourth, the stateful design increases implementation complexity. Session management, context synchronization, and distributed scaling require careful design [42], [44]. Compared to stateless REST systems, MCP may require more operational effort, especially in high-load or low-latency environments. Finally, MCP is still an emerging standard, and its long-term adoption depends on ecosystem support, platform compatibility, and stable implementations.

4.10. Future Directions and Research Opportunities

The development of agentic AI shows a gap between advances in autonomous reasoning and the infrastructure required to support these systems at scale. Frameworks such as the Model Context Protocol (MCP) address context management and interoperability, but several challenges remain. One area is context representation and management. Although MCP supports stateful communication, questions remain about how long-term memory should be stored and maintained during extended tasks. Research is needed to balance reasoning quality, computational cost, and privacy, especially in multi-agent and long-term settings.

Another challenge is integration with data infrastructure. Many existing data platforms were not designed for the continuous and semantic interaction required by autonomous agents. Research should explore hybrid data systems that combine vector databases, knowledge graphs, streaming systems, and symbolic data under interoperability layers such as MCP. It is also important to study how these systems affect performance, latency, and reliability. Multi-agent coordination and governance is also another issue. While MCP enables shared context and tool access, further work is needed on conflict resolution, role management, trust, and accountability among agents. Research should examine protocol-level mechanisms that support coordination and predictable group behavior. Performance evaluation is also limited. Few benchmarks measure contextual consistency, failure recovery, adaptability, and integration cost. Developing evaluation frameworks for agentic AI infrastructures, including MCP-based systems, would support practical deployment.

Security and governance require attention. Because MCP allows dynamic tool access, research is needed on access control, monitoring, and policy enforcement, especially in regulated domains such as healthcare and finance. Finally,

interoperability standards must evolve as agent capabilities expand. Future work should study how protocols like MCP can support new agent types, including self-improving agents, embodied systems, and human–AI collaboration. Addressing these issues will support the reliable deployment of agentic AI systems.

5. Conclusion

This review examined agentic AI as a continuation of earlier agent-based artificial intelligence research. Modern systems build on ideas from rational agents, BDI architectures, planning systems, multi-agent systems, and reinforcement learning. Although current agentic systems use neural and probabilistic methods instead of symbolic logic, they retain core structures such as perception–action loops, goal management, planning, coordination, and feedback. The analysis showed that agentic AI depends on modular components working together, including perception, memory, reasoning, execution, and reflection. To support large-scale deployment, these systems require infrastructure that maintains context and reduces integration complexity. The Model Context Protocol (MCP) addresses this need by standardizing tool access, stateful sessions, and capability discovery. However, MCP operates at the interoperability layer and does not define reasoning models or coordination semantics. It improves integration but does not ensure consistent planning, negotiation, or governance. Its stateful design also increases implementation and security requirements. Future research should focus on improving coordination mechanisms, developing evaluation benchmarks, strengthening security controls, and integrating interoperability standards with cognitive reasoning frameworks. Progress in agentic AI will depend not only on model capability but also on reliable infrastructure, governance, and system-level integration.

References

[1] Dheer Toprani, and Vijay K. Madisetti, "LLM Agentic Workflow for Automated Vulnerability Detection and Remediation in Infrastructure-as-Code," *IEEE Access*, vol. 13, pp. 69175–69181, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[2] Ilias Chatzistefanidis, Andrea Leone, and Navid Nikaein, "Maestro: LLM-Driven Collaborative Automation of Intent-Based 6G Networks," *IEEE Networking Letters*, vol. 6, no. 4, pp. 227–231, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[3] Nalan Karunanayake, "Next-generation Agentic AI for Transforming Healthcare," *Informatics and Health*, vol. 2, no. 2, pp. 73–83, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[4] Joon Yul Choi, and Tae Keun Yoo, "Transforming Clinical Medicine with Multimodal Artificial Intelligence, Agentic Systems, and the Model-context Protocol : A Perspective on Future Directions," *Discover Health Systems*, vol. 5, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[5] Mohamad Abou Ali, Fadi Dornaika, and Jinan Charafeddine, "Agentic AI: A Comprehensive Survey of Architectures, Applications, and Future Directions," *Artificial Intelligence Review*, vol. 59, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[6] IBM, What is agentic AI?. [Online]. Available: https://www.ibm.com/think/topics/agentic-ai

[7] Vallikranth Ayyagari, "Model Context Protocol for Agentic AI : Enabling Contextual Interoperability Across Systems," *International Journal of Computational and Experimental Science and Engineering*, vol. 11, no. 3, 2025. [CrossRef] [Publisher Link]

[8] Micaela Kaplan, How Model Context Protocol Connects LLMs to the Real World, Label Studio. [Online]. Available: https://labelstud.io/blog/how-model-context-protocol-connects-llms-to-the-real-world/

[9] Manas Deepak Patil, and Virag Vijay Lokhande, "Model Context Protocol MCP Enabling Scalable Ai Data Integration," *International Journal for Multidisciplinary Research*, vol. 7, no. 2, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[10] Aske Plaat et al., "Agentic Large Language Models, A Survey," *Journal of Artificial Intelligence Research*, vol. 84, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[11] Johannes Schneider, "Generative to Agentic AI: Survey, Conceptualization, and Challenges," *arXiv:2504.18875*, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[12] Deepak Bhaskar Acharya, Karthigeyan Kuppan, and B. Divya, "Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey," *IEEE Access*, vol. 13, pp. 18912–18936, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[13] Mourad Gridach et al., "Agentic AI for Scientific Discovery: A Survey of Progress, Challenges, and Future Directions," *arXiv:2503.08979*, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[14] Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee, "AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges," *Information Fusion*, vol. 126, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[15] Soodeh Hosseini, and Hossein Seilani, "The Role of Agentic AI in Shaping a Smart Future: A Systematic Review," *Array*, vol. 26, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[16] Baoyu Liang, Yuchen Wang, and Chao Tong, "AI Reasoning in Deep Learning Era: From Symbolic AI to Neural–Symbolic AI," *Mathematics,* vol. 13, no. 11, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[17] T. Nithya et al., "A Comprehensive Survey of Machine Learning: Advancements, Applications, and Challenges," *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Thomas Rincy N, and Roopam Gupta, "A Survey on Machine Learning Approaches and Its Techniques," *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[19] William Grant Hatcher, and Wei Yu, "A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends," *IEEE Access*, vol. 6, pp. 24411–24432, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[20] Md Zahangir Alom et al., "A State-of-the-Art Survey on Deep Learning Theory and Architectures," *Electronics,* vol. 8, no. 3, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[21] Sandeep Singh Sengar et al., "Generative Artificial Intelligence: A Systematic Review and Applications," *Multimedia Tools and Applications*, vol. 84, pp. 23661–23700, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[22] Tam Sakirin, and Siddartha Kusuma, "A Survey of Generative Artificial Intelligence Techniques," Babylonian Journal of Artificial Intelligence, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[23] Francesco Piccialli et al., "AgentAI: A Comprehensive Survey on Autonomous Agents in Distributed AI for Industry 4.0," *Expert Systems with Applications*, vol. 291, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[24] Andreas Brännström et al., "A Formal Framework for Designing Boundedly Rational Agents," *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, vol. 3, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[25] Prashant Gupta, Rational Agents for Artificial Intelligence, Medium, 2017. [Online]. Available: https://medium.com/hackernoon/rational-agents-for-artificial-intelligence-caf94af2cec5

[26] Laurent Frering, Gerald Steinbauer-Wagner, and Andreas Holzinger, "Integrating Belief-Desire-Intention Agents with Large Language Models for Reliable Human–robot Interaction and Explainable Artificial Intelligence," *Engineering Applications of Artificial Intelligence*, vol. 141, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[27] Geylani Kardas, Baris Tekin Tezel, and Moharram Challenger, "Domain-specific Modelling Language for Belief–Desire–Intention Software Agents," *IET Software*, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[28] Diego Aineto, Sergio Jiménez, and Eva "Learning STRIPS Action Models with Classical Planning," *arXiv:1903.01153*, 2019. [CrossRef] [Publisher Link]

[29] Xenija Neufeld, Sanaz Mostaghim, and Sandy Brand, "A Hybrid Approach to Planning and Execution in Dynamic Environments Through Hierarchical Task Networks and Behavior Trees," *Fourteenth Artificial Intelligence and Interative Digital Entertainment Conference*, vol. 14, no. 1, pp. 201–207, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[30] Sofia Kouah et al., "Internet of Things-Based Multi-Agent System for the Control of Smart Street Lighting," *Electronics*, vol. 13, no. 18, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[31] Jiaxin Liang et al., "A Review of Multi-Agent Reinforcement Learning Algorithms," *Electronics*, vol. 14, no. 4, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[32] Shuo Yang et al., "Reinforcement Learning Agents Playing Ticket to Ride–A Complex Imperfect Information Board Game With Delayed Rewards," *IEEE Access*, vol. 11, pp. 60737–60757, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[33] Ajay Bandi et al., "The Rise of Agentic AI : A Review of Definitions, Frameworks, Architectures, Applications, Evaluation Metrics, and Challenges," *Future Internet*, vol. 17, no. 9, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[34] Markets and Markets, AI Agents Market Size, Share, Growth and Lates Trends, 2025. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/ai-agents-market-15761548.html

[35] AI Agents Market (2026 - 2033), Grand View Research. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/ai-agents-market-report

[36] Venus Garg, "Designing the Mind: How Agentic Frameworks Are Shaping the Future of AI Behavior," *Journal of Computer Science and Technology Studies*, vol. 7, no. 5, pp. 182–193, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[37] Mohammed Salah et al., "Agentic Artificial Intelligence in Public Administration: Foundations, Applications, and Governance Mohammed Salah," *SSRN*, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[38] Yi Yu et al., "Agentic Memory: Learning Unified Long-Term and Short-Term Memory Management for Large Language Model Agents," *arXiv: 2601.01885*, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[39] Gokcer Belgusen, Memory Types in Agentic AI: A Breakdown, Medium, 2025. [Online]. Available: https://medium.com/@gokcerbelgusen/memory-types-in-agentic-ai-a-breakdown-523c980921ec

[40] Mohan Varthakavi, Reimagining Data Architecture for Agentic AI, DataVarsity, 2025. [Online]. Available: https://www.dataversity.net/articles/reimagining-data-architecture-for-agentic-ai/#:~:text=would%20likely%20miss.-

[41] Manish Sood, "Agentic AI is Forcing a Shift from Application-first to Data-first Architecture, RELTIO, 2026. [Online]. Available: https://www.reltio.com/resources/blog/agentic-ai-is-forcing-a-shift-from-application-first-to-data-first-architecture/#:~:text=It

[42] Xinyi Hou et al., "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions," *ACM Transactions of Software Engineering and Methodology*, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[43] Anna Gutowska, What is Model Context Protocol (MCP)?, IBM. [Online]. Available: https://www.ibm.com/think/topics/model-context-protocol

[44] Padmanabhan Venkiteela, "The New Interoperability Paradigm : Model Context Protocol (MCP), APIs, and the Future of Agentic AI," *Computer Fraud and Security*, vol. 2025, no. 1, pp. 1259–1271, 2025. [Google Scholar] [Publisher Link]

[45] Aniket P. Kakde et al., "Advancing Agentic AI through Communication Protocols," *International Journal of Scientific Research in Science and Technology*, vol. 12, no. 5, pp. 299–308, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[46] Manoj Kumar Rath, "Adaptive Model Context Protocols for Trustworthy and Secure Agentic AI Systems," *International Journal of Computer Technology and Electronics Communication*, vol. 8, no. 5, pp. 11444–11456, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[47] Codiste, Model Context Protocol (MCP) vs Traditional APIs: Key Differences and Use Cases, Medium. [Online]. Available: https://medium.com/@nishantbj/model-context-protocol-mcp-vs-traditional-apis-key-differences-and-use-cases-24dc023f16f2

[48] Herman Errico, Jiquan Ngiam, and Shanita Sojan, "Securing the Model Context Protocol (MCP): Risks, Controls, and Governance," *arXiv: 2511.20920*, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[49] Sonu Kumar et al., "MCP Guardian: A Security-First Layer for Safeguarding MCP-Based AI," *Computer Science and Information Technology*, pp. 107–121, 2025. [CrossRef] [Publisher Link]

[50] Grigoris Ntousakis et al., "Securing MCP-Based Agent Workflows," *Proceedings of the 4th Workshop on Practical Adoption Challenges of ML for Systems*, pp. 50-55, 2025. [CrossRef] [Google Scholar] [Publisher Link]