

From Foundations to Autonomous Ecosystems: A Comprehensive Survey of Agentic AI Architectures, Protocols, Socio-Technical Dynamics, and Future Trajectories.

Student, **Mr. Vipul Gangwani**

School of sciences, KK Modi University, Durg

HOD, **Dr. Mohammed Bakhtawar Ahmed**

School of sciences, KK Modi University, Durg

Abstract

The paradigm shift from passive Generative AI to proactive Agentic AI marks a pivotal evolution in artificial intelligence, transcending static content generation to enable autonomous, goal-directed task execution within dynamic environments. This comprehensive survey systematically bridges the historical divide between classical symbolic architectures and modern neural-generative lineages, formalizing a unified taxonomy that resolves conceptual retrofitting. By leveraging the Cognitive Architectures for Language Agents (CoALA) framework and a 10-dimension Cognitive Faculty matrix, we deconstruct the operational mechanics of multi-agent ecosystems across persistent memory hierarchies, interleaved reasoning-and-acting loops (e.g., ReAct, Tree of Thoughts), and dynamic tool utilization. Furthermore, we investigate the critical role of standardized inter-agent communication protocols (MCP, A2A, ACP, ANP, Agora) in facilitating robust orchestration within frameworks like AutoGen and CrewAI. Transitioning from theoretical foundations to practice, we provide deep empirical case studies in high-stakes domains, including 6G intelligent communications, automated software engineering, and clinical decision support. Crucially, this survey addresses the profound socio-technical implications of delegated autonomy through the MAD-BAD-SAD framework, analyzing emerging threat surfaces such as prompt injection, memory poisoning, and the "lethal trifecta." Finally, we critically evaluate the necessary shift from outcome-based to process-oriented benchmarking and outline future trajectories—encompassing lifelong learning, Green Agentic AI, and neuro-symbolic convergence—charting the rigorous path from isolated algorithmic agents toward Artificial General Intelligence (AGI).

Keywords

Agentic AI, Artificial General Intelligence (AGI), Cognitive Architectures, Large Language Models (LLMs), Multi-Agent Systems (MAS), Interoperability Protocols, Socio-Technical Systems, Trustworthy AI.

I. INTRODUCTION & HISTORICAL EVOLUTION

The landscape of artificial intelligence (AI) is undergoing a profound paradigm shift, transitioning from the development of passive, task-specific generative models to the engineering of autonomous, goal-directed systems that exhibit genuine agency. In recent years, Large Language Models (LLMs) have evolved from serving as conversational interfaces to functioning as the central cognitive engines within complex computational workflows. This transition marks the dawn of Agentic AI, a paradigm wherein intelligent systems proactively perceive environments, reason over extended time horizons, orchestrate tool usage, and collaborate within multi-agent ecosystems to achieve high-level objectives. However, the rapid pace of this advancement has precipitated a fragmented understanding of agentic architectures, necessitating a rigorous taxonomy to differentiate modern neural systems from historical symbolic models.

A. The Evolution of Agency in Artificial Intelligence

The pursuit of computational agency is not a novel endeavor; rather, it is the culmination of decades of research spanning several distinct eras, each defined by specific architectural paradigms and cognitive representations.

1. The Symbolic/Classical Era (1950s–1980s): The foundational ambition of artificial intelligence was grounded in explicit logic, human-crafted rules, and symbolic representation. During this period, intelligence was conceived as a top-down, deductive process. Systems such as MYCIN and DENDRAL operated on meticulously designed knowledge bases to perform expert tasks. Agency in this era was modeled through cognitive architectures like SOAR and Belief-Desire-Intention (BDI) frameworks, where intentions served to maintain long-term commitments and guide algorithmic planning.

2. The Machine Learning and Deep Learning Eras (1980s–2010s): The field gradually shifted away from hard-coded logic toward statistical learning, marking the Machine Learning era. Algorithms like Support Vector Machines (SVMs) learned representations from data, though they remained highly dependent on human feature engineering. The subsequent Deep Learning era, catalyzed by increased computational power and massive datasets, utilized Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to automatically learn hierarchical representations. While these architectures shattered barriers in perception—such as image classification and speech recognition—they functioned primarily as sophisticated pattern classifiers rather than autonomous, goal-seeking agents.

3. The Generative AI Breakthrough (2014–2022): The introduction of the Transformer architecture in 2017 revolutionized AI by enabling the scaling of Large Language Models (LLMs) like GPT and BERT. Generative AI systems demonstrated unprecedented capabilities in synthesizing coherent text, code, and multimedia. However, despite their impressive fidelity, these models were fundamentally reactive. They operated exclusively on user-provided prompts, lacking persistent memory, internal state management, or intrinsic goal-following mechanisms. Generative AI acted as an assistant requiring granular, step-by-step instructions rather than an autonomous entity.

4. The Agentic AI Era (2022–Present): The current frontier harnesses the statistical reasoning capabilities of LLMs to achieve action and autonomy. By embedding LLMs within closed-loop architectures that include memory modules, API tool access, and recursive planning functions, static models are transformed into dynamic agents. This era is characterized by the orchestration of multiple specialized agents communicating and collaborating to resolve tasks that exceed the cognitive capacity of a single model.

B. The Definitional Divide: AI Agents vs. Agentic AI

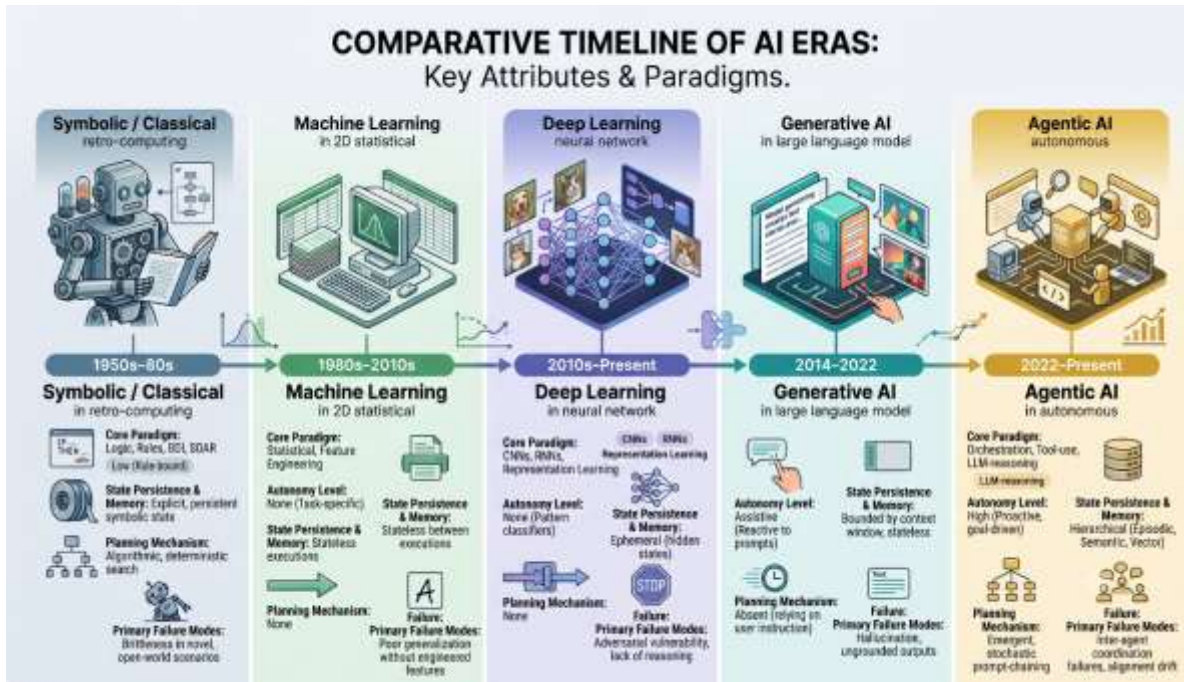
A critical issue in contemporary AI literature is "conceptual retrofitting"—the pervasive misapplication of classical symbolic frameworks, such as BDI or explicit Perceive-Plan-Act-Reflect (PPAR) loops, to describe modern neural systems. Modern LLM-based agents operate on principles of stochastic generation and prompt-driven orchestration, which are fundamentally incompatible with the deterministic symbol manipulation of classical AI. To resolve this, it is imperative to formally delineate the boundaries between "AI Agents" and "Agentic AI."

AI Agents refer to single-entity, autonomous software systems designed for goal-directed task execution within bounded environments. They are modular and rely on LLMs or Large Image Models (LIMs) as core reasoning engines to perceive

inputs, utilize localized external tools, and perform sequential reasoning. Their autonomy is often restricted to well-defined, isolated tasks.

Agentic AI, conversely, represents a broader architectural approach centered on the orchestration of multi-agent systems (MAS). In Agentic AI ecosystems, multiple specialized agents (e.g., a planner, a researcher, a coder, and a reviewer) collaborate dynamically, sharing memory states and dividing tasks to pursue highly complex, open-ended objectives. Where an AI Agent is a single sophisticated worker, Agentic AI is the coordinated enterprise.

Table I: Evolution of AI Eras and Agentic Capabilities



C. Research Scope, Contributions, and Methodology

This survey systematically evaluates the socio-technical, architectural, and operational dimensions of Agentic AI. By mapping the transition from classical frameworks to multi-agent ecosystems, this work contributes a unified Dual-Paradigm Taxonomy, deeply analyzes inter-agent communication protocols, and evaluates real-world applications across high-stakes domains (e.g., 6G networks, healthcare). Furthermore, it addresses the ethical and security challenges encapsulated in the MAD-BAD-SAD framework, concluding with a rigorous evaluation of process-oriented benchmarking metrics.

II. THEORETICAL FOUNDATIONS & TAXONOMY

To avoid the pitfalls of conceptual retrofitting, the theoretical underpinnings of Agentic AI must be analyzed through a framework that respects the distinct evolutionary tracks of artificial intelligence. Furthermore, measuring the progress of these systems toward Artificial General Intelligence (AGI) requires rigorous cognitive and performance-based benchmarks.

A. Dual-Paradigm Taxonomy of Agentic AI

Agentic systems can be categorized into two fundamentally distinct lineages: the Symbolic/Classical Lineage and the Neural/Generative Lineage.

1. The Symbolic/Classical Lineage: This paradigm is defined by explicit logic, algorithmic decision-making, and persistent symbolic states. Foundational constructs include Markov Decision Processes (MDPs) and Partially Observable MDPs (POMDPs), which provide mathematical scaffolding for agent planning under uncertainty by utilizing probabilistic belief states. In this lineage, cognitive architectures such as Belief-Desire-Intention (BDI) and SOAR explicitly model human executive control, maintaining working memory and formal goal stacks. Planning is a top-down, deterministic process. However, these systems scale poorly to the high-dimensional, unstructured data of the real world, rendering them highly brittle outside constrained environments.

2. The Neural/Generative Lineage: In direct opposition, the neural lineage derives agency from statistical learning and the emergent reasoning capabilities of Large Language Models (LLMs). This paradigm discards hand-crafted logic in favor of bottom-up, prompt-driven orchestration. Agency is not achieved through a formal BDI loop, but through mechanisms like prompt chaining, context management, and external tool integration. Frameworks such as LangChain and AutoGen orchestrate stochastic generation across networks of LLMs. The defining characteristic of this lineage is its extreme adaptability to unstructured data; however, it suffers from opacity, making formal verification and logical soundness difficult to guarantee.

B. Levels of AGI Framework

As Agentic AI architectures grow in sophistication, their capabilities increasingly approximate attributes of Artificial General Intelligence (AGI). To operationalize the measurement of progress toward AGI, Morris et al. propose a matrixed leveling framework based on two critical dimensions: **Performance** (depth of capability relative to human adults) and **Generality** (breadth of tasks the system can handle).

The ontology categorizes systems into six performance levels:

Level 0: No AI (e.g., simple calculator software).

Level 1: Emerging (Equal to or somewhat better than an unskilled human). Contemporary frontier LLMs such as ChatGPT, Claude, and Llama 2 currently reside at this level (Emerging AGI), demonstrating broad but often unreliable competence across many tasks.

Level 2: Competent (At least 50th percentile of skilled adults).

Level 3: Expert (At least 90th percentile of skilled adults).

Level 4: Exceptional (At least 99th percentile of skilled adults).

Level 5: Superhuman (Outperforms 100% of humans). While Superhuman Narrow AI exists (e.g., AlphaFold for protein folding, AlphaZero for chess), Artificial Superintelligence (ASI) remains unachieved.

A critical assertion of this framework is that AGI must be defined by **capabilities** (what the system can accomplish) rather than **processes** (how the system thinks), rendering debates over machine consciousness orthogonal to AGI benchmarking.

C. Cognitive Taxonomy for Agentic Systems

To achieve higher levels of AGI, an agentic system must demonstrate a comprehensive array of human-like cognitive abilities. Burnell et al. outline a rigorous Cognitive Taxonomy that deconstructs general intelligence into 10 distinct cognitive faculties.

Current Agentic AI frameworks excel in **Generation** and **Semantic Memory** but exhibit profound vulnerabilities in **Metacognition**, **Executive Functions** (particularly long-horizon planning), and **Social Cognition**. The architectural evolution necessary to bridge these gaps forms the core of the subsequent sections of this survey.

III. ARCHITECTURAL MODELS & COGNITIVE FRAMEWORKS

The transition from isolated, prompt-driven generative models to autonomous, goal-directed systems necessitates the implementation of robust cognitive architectures. These frameworks structure the internal reasoning, memory management, and action execution of Large Language Models (LLMs), effectively transforming them from stateless text generators into persistent, interactive agents.

A. Cognitive Architectures for Language Agents (CoALA)

To systematize the design of language agents, the Cognitive Architectures for Language Agents (CoALA) framework maps the rich history of symbolic AI and cognitive science onto modern LLM-based systems. CoALA conceptualizes the LLM as the core computational unit embedded within a broader architectural loop consisting of distinct memory modules, a structured action space, and an iterative decision-making procedure.

1. The Memory Hierarchy: Language models are inherently stateless; however, CoALA posits that true agency requires the explicit organization of information into a multi-tiered memory hierarchy:

Working Memory: This short-term memory serves as the central hub of the agent, maintaining active variables for the current decision cycle. It houses recent perceptual inputs, active goals, and intermediate reasoning outputs. During execution, the LLM's input is dynamically synthesized from working memory, and its parsed outputs are written back to it.

Episodic Memory: This long-term memory module stores sequences of the agent's past behaviors, environmental event flows, and historical task trajectories. During planning, agents can retrieve these episodes to inform current decision-making.

Semantic Memory: Acting as the agent's factual knowledge base, semantic memory stores world knowledge. While traditional Retrieval-Augmented Generation (RAG) uses read-only external databases, CoALA agents can actively write newly synthesized inferences to semantic memory, enabling continuous knowledge accumulation.

Procedural Memory: This encompasses both the implicit knowledge stored within the LLM's parametric weights and the explicit, human-authored agent code (e.g., prompt templates, API parsing scripts). Procedural memory dictates **how** the agent operates and interacts with its other memory modules.

2. Structuring the Action Space: An agent's capability is defined by its action space, which CoALA bifurcates into internal and external categories:

Internal Actions: These actions interface exclusively with the agent's memory. They are subdivided into **retrieval** (reading from long-term to working memory), **reasoning** (using the LLM to process working memory and generate new insights), and **learning** (writing newly generated information from working memory back to long-term storage).

External Actions (Grounding): These procedures execute actions that impact the external world, effectively translating the agent's internal text-based decisions into digital API calls, dialogue with humans, or physical robotic motor commands.

3. The Decision-Making Loop: The top-level agent program operates as a continuous decision cycle. In the Planning stage, the agent utilizes reasoning and retrieval to propose multiple candidate actions, evaluate their expected utility

(often using the LLM to simulate outcomes), and select the optimal path. In the Execution stage, the selected internal learning action or external grounding action is applied, triggering a new observation from the environment and initiating the next cycle.

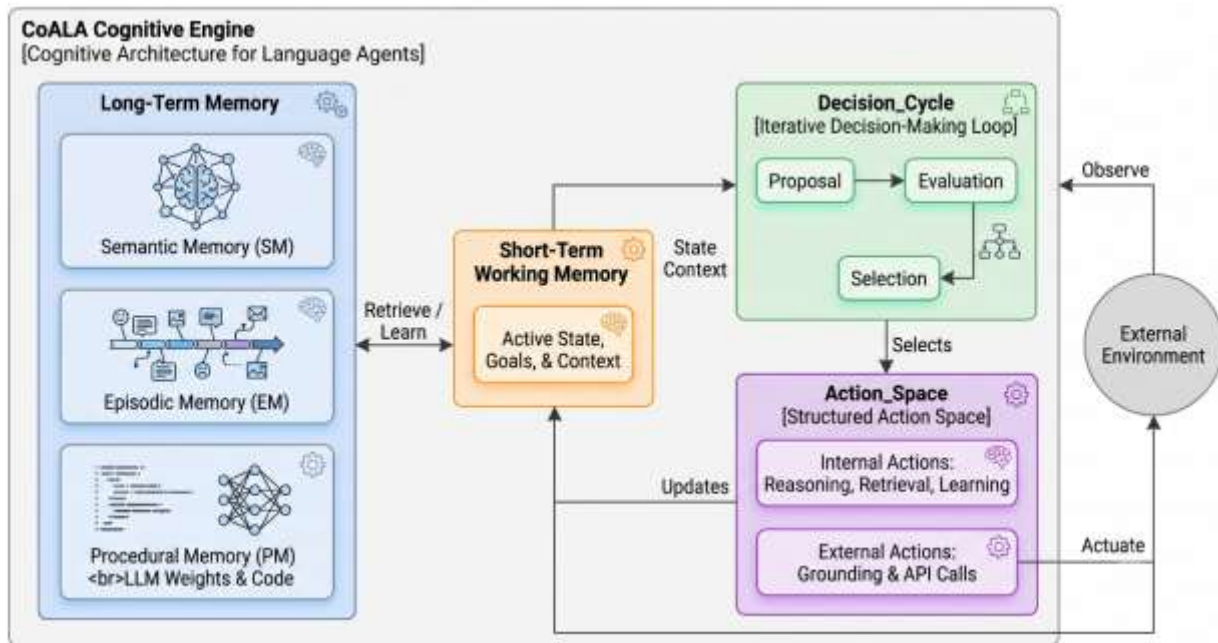


Diagram 1: The CoALA-inspired Agentic AI Cognitive Engine mapping the interplay between memory hierarchies, the iterative decision cycle, and the structured action space.

B. Advanced Reasoning and Planning Paradigms

To execute complex, long-horizon tasks, Agentic AI systems employ structured reasoning paradigms that significantly transcend basic prompt engineering.

- Interleaved Reasoning and Acting (ReAct):** The ReAct framework addresses the frequent hallucination and error propagation observed in pure Chain-of-Thought (CoT) reasoning. ReAct prompts the LLM to generate alternating reasoning traces ("Thoughts") and task-specific external interactions ("Actions"), followed by environmental feedback ("Observations"). This synergy ensures that internal planning remains rigorously grounded in external reality, allowing the agent to adjust its plans dynamically when confronting exceptions.
- Tree of Thoughts (ToT), Graph of Thoughts (GoT), and MCTS:** Recognizing the limitations of linear, single-path reasoning, advanced architectures simulate classical search algorithms. The Tree of Thoughts (ToT) paradigm enables the LLM to explore multiple reasoning paths concurrently via Breadth-First Search (BFS) or Depth-First Search (DFS). The model self-evaluates intermediate "thoughts," pruning paths that are unlikely to yield success. Graph of Thoughts (GoT) extends this by allowing cyclic reasoning, where thoughts can be dynamically merged and refined as nodes within a graph structure. Furthermore, Monte Carlo Tree Search (MCTS) has been integrated into agentic reasoning to incrementally build search trees and rigorously evaluate simulated rollouts, optimizing the agent's long-term action selection.

C. Agentic Retrieval-Augmented Generation (Agentic RAG)

Classical RAG systems perform a static, single-step retrieval from a vector database to augment an LLM's context window. Agentic RAG transforms this into a dynamic, multi-agent process. Rather than executing a predefined search,

specialized retriever agents autonomously determine *which* sources to query, *evaluate* the relevance and factual integrity of the retrieved information, and decide if iterative or reformulated retrieval is necessary. By utilizing persistent memory and interactive feedback, Agentic RAG ensures that external knowledge is seamlessly and accurately integrated into the system's overarching multi-step planning process, drastically reducing hallucinations.

D. Vibe Coding vs. Agentic Coding

The advent of agentic capabilities has catalyzed a profound paradigm shift in software engineering, delineated by the divergence between "Vibe Coding" and "Agentic Coding".

1. Vibe Coding (Intuition-Driven Co-Creation): Vibe coding relies on a conversational, prompt-response loop where the human developer acts as a creative director. The developer articulates high-level intent (the "vibe"), and the LLM acts as a rapid co-pilot to generate code snippets. This paradigm is highly dependent on five cognitive skills from the human: Strategic Problem Formulation, Architectural Awareness, Checkpointing, Collaborative Debugging, and Context Provision. The execution, integration, and testing of the code remain entirely manual and human-driven, making it ideal for rapid prototyping but brittle for enterprise-scale systems.

2. Agentic Coding (Autonomous Software Engineering): Agentic coding delegates the cognitive and operational burdens to autonomous agents. The developer shifts to a supervisory role, setting high-level strategic objectives. The agentic system autonomously decomposes these goals into sub-tasks, utilizes embedded tools (e.g., compilers, Git), executes code within secure, containerized sandboxes, and relies on automated testing pipelines to iteratively debug its own code. By embedding safety constraints and rollback mechanisms, agentic coding transforms software engineering into an autonomous, closed-loop pipeline capable of large-scale legacy migration and Continuous Integration/Continuous Deployment (CI/CD) orchestration.

IV. AGENT COMMUNICATION PROTOCOLS & MULTI-AGENT ORCHESTRATION

The true potential of Agentic AI is realized when multiple specialized agents coordinate to solve complex problems. However, governing these multi-agent systems (MAS) requires robust orchestration frameworks and standardized communication protocols to prevent emergent instability and semantic fragmentation.

A. Multi-Agent Orchestration Frameworks

Recent literature highlights several predominant frameworks that structure inter-agent coordination, effectively replacing centralized scheduling algorithms with dynamic, prompt-driven process management.

1. AutoGen (Conversational Orchestration): AutoGen conceptualizes multi-agent workflows as autonomous conversations. It utilizes customizable 'ConversableAgent' modules that can seamlessly integrate LLMs, human inputs, and tools. Through "conversation programming," AutoGen relies on a 'GroupChatManager' to dynamically select speakers based on the conversational context, allowing agents to collaboratively debate, write code, and execute tasks without rigid, predefined communication topologies.

2. CrewAI (Role-Based Hierarchical Delegation): CrewAI orchestrates agents by explicitly assigning distinct professional roles, goals, and backstories to a "crew" of agents. By mimicking corporate structures, an orchestrator (or manager agent) delegates sub-tasks to specialized agents (e.g., a data processor or a report writer), ensuring clear behavioral boundaries and providing an auditable trail of agent actions.

3. LangGraph (State-Machine Workflows): LangGraph imposes rigorous structure on multi-agent collaboration by modeling interactions as cyclic state machines. Agents and tools serve as nodes within a graph, while edges define the conditional logic governing the workflow. This architecture is highly effective for tasks requiring strict temporal consistency, persistent state management across extended interactions, and compositional flows.

B. Standardized Communication & Interoperability Protocols

As the ecosystem of Agentic AI rapidly expands, the reliance on proprietary, ad-hoc API integrations creates severe interoperability, security, and scalability bottlenecks. To facilitate decentralized agent markets and cross-vendor collaboration, researchers have proposed several standardized Agent Communication Protocols.

1. **Model Context Protocol (MCP):** Initially designed for structured tool calls, MCP operates on a client-server model using JSON-RPC formats. It standardizes how agents discover and interact with local and remote data sources, offering secure schema validation. While primarily tool-focused, it supports hierarchical inter-agent delegation.
2. **Agent-to-Agent Protocol (A2A):** Developed by Google, A2A explicitly targets cross-agent collaboration. It utilizes formal constructs such as "Agent Cards" for capability discovery, "Task Objects" for goal coordination, and "Artifacts" for standardized outputs. Operating via JSON-LD over HTTP, A2A enables robust memory management and enterprise-scale agent orchestration.
3. **Agent Communication Protocol (ACP) & Agent Network Protocol (ANP):** ACP utilizes RESTful APIs and structured JSON messages to encode specific actions, goals, and intents, making it transport-agnostic and highly scalable. ANP focuses on decentralized environments (Web3), incorporating Decentralized Identifiers (DIDs) and organizing agent communication around strict lifecycle stages (creation, operation, update, termination).
4. **Agora:** Functioning as a meta-coordination layer, Agora integrates multiple underlying protocols (including MCP and ACP) through Protocol Documents (PDs). These machine-interpretable specifications allow agents to negotiate their communication methods dynamically using natural language, making Agora uniquely suited for highly heterogeneous multi-agent environments.

TABLE II: COMPARATIVE MATRIX OF MODERN AGENT COMMUNICATION PROTOCOLS

	MCP	ACP	A2A	ANP	Agora
Message Format	JSON-RPC	JSON-LD	JSON-LD + NLP	JSON-RPC / HTTP	Protocol Docs (PD) + NLP
Semantics	Custom performatives	Goal-oriented messages (goal, action)	Custom performatives	Protocol Documents	Protocol Documents
Discovery	Manual	Registry & metadata (agent.yml)	Agent Card (JSON-LD)	Decentralized Identifiers	Exchanging PDs via NLP
Transport Layer	HTTP, Stdio, SSE	HTTP	HTTP, optional SSE	HTTP with JSON-LD	HTTP with PD
Primary Use Case	LLM-tool integration	Cross-agent collaboration	Enterprise orchestration	Decentralized markets	Heterogeneous MAS environments
Framework Adopters	LangChain, Agno	AutoGen, LangGraph	AutoGen, CrewAI	AutoGen, CrewAI	Agora, Semantic Kernel

The transition of Agentic AI from theoretical cognitive frameworks to real-world deployment requires domain-specific optimizations to handle immense complexity, strict safety constraints, and dynamic environmental feedback. Empirical case studies across software engineering, telecommunications, and healthcare reveal how agentic ecosystems operate under practical stress, highlighting both their unprecedented capabilities and current limitations.

A. Automated Bug Fixing and Software Engineering

The domain of automated program repair has been revolutionized by LLM-based agents capable of autonomously exploring code repositories, reproducing bugs, and generating validating patches. The performance of these agents is rigorously evaluated on benchmarks such as SWE-bench Verified, a human-annotated dataset of 500 complex real-world GitHub issues across popular Python repositories. Unlike traditional static code analyzers, Agentic AI systems in this

domain dynamically interact with local environments (via bash terminals), retrieve repository contexts, edit code, and execute unit tests (Fail-to-Pass or F2P tests) to iteratively validate their repair strategies.

1) Empirical Analysis of SWE-bench Verified: An analysis of the top-performing bug-fixing agent ecosystems reveals significant variations in architectural efficacy. W&B Programmer emerged as the state-of-the-art system, successfully resolving 64.6% of the 500 SWE-bench Verified cases (323 instances). Its success is attributed to its utilization of the OpenAI o1 model operating in a high-reasoning mode, coupled with a GPT-4o-based memory compression component that distills the agent's historical context window, thereby preventing context degradation during extended troubleshooting. Furthermore, W&B Programmer employs a multi-rollout strategy, generating five parallel trajectories per instance and utilizing an algorithmic "crosscheck" step to dynamically select the optimal patch.

Conversely, alternative architectures demonstrate value through orthogonal approaches. The Learn-by-interact framework, for example, successfully resolved 12 unique cases that no other top-tier system (including W&B Programmer, Devlo, and Blackbox AI) could fix. Instead of relying solely on real-time reasoning, Learn-by-interact utilizes an offline mode to gather high-quality interaction trajectories, strategically integrating historical debugging patterns into its active prompts via in-context learning. This proves that experiential knowledge accumulation—a form of long-term episodic memory integration—can empower agents to solve edge-case defects that evade standard deductive reasoning pipelines.

2) Fault Localization (FL): File-Level vs. Code Symbol-Level: The prerequisite for automated bug fixing is accurate Fault Localization (FL). Empirical studies measuring FL accuracy demonstrate a strong positive correlation between fine-grained localization and ultimate patch success. While systems like W&B Programmer exhibit high proficiency in file-level FL (successfully identifying at least one buggy file in 448 out of 500 cases), code symbol-level localization (pinpointing specific classes, functions, or methods) remains a critical bottleneck. W&B Programmer localized buggy code symbols in only 396 cases, highlighting that while agents can navigate macro-repository structures effectively, their micro-level semantic precision requires further optimization.

3) Bug Reproduction and Patch Validation: The capacity of an agent to autonomously write bug reproduction scripts is paramount. An empirical test deploying a customized "RepoFixer" dual-agent architecture (comprising a Searcher and a Fixer) demonstrated that 76.8% of the reproduction scripts generated by the agent successfully yielded state divergence before and after applying the golden patch. The quality of the initial human issue description directly impacts this capability; issues containing explicit, reproducible examples led to an 80.4% success rate in agent-driven reproduction, dropping to 64% when the issue lacked sufficient context. This underscores that human-agent interaction and the quality of natural language specifications are critical variables in autonomous software engineering.

B. Future Intelligent Communications and 6G Networks

As wireless networks evolve toward the 6G standard, they demand unprecedented levels of dynamic adaptability, ultra-low latency, and autonomous resource management. Traditional rule-based network algorithms are ill-equipped to handle the volatility of space-air-ground integrated networks. Consequently, the telecommunications sector is witnessing a paradigm shift from utilizing Large AI Models (LAMs) as passive analytical tools to deploying Agentic AI systems as active network orchestrators.

1) LAMs vs. Agentic AI in Network Architecture: In the context of 6G, LAMs function primarily as Data Generators (synthesizing high-quality Channel State Information) and Knowledge Organizers (semantic encoding). However, Agentic AI introduces autonomous decision-making loops. Agentic frameworks function as Task Schedulers and System Designers, actively comprehending complex network instructions, decomposing them into sub-tasks, and orchestrating execution across edge devices and cloud servers.

2) Agentic Semantic Communication Codecs: Semantic communication transcends traditional Shannon-Weaver bit-transmission by focusing on the "meaning" of the data conveyed. Agentic AI optimizes this through frameworks like Agent-driven Generative Semantic Communication (A-GSC). In this architecture, agents utilize Deep Reinforcement

Learning (DRL) to adaptively adjust semantic sampling, extraction, and compression rates in real-time, responding dynamically to varying channel noise and bandwidth constraints. Furthermore, Multi-User Generative Semantic Communication (M-GSC) utilizes orchestrator agents to decompose complex multi-user tasks, normalizing semantic representations and managing shared knowledge bases to optimize spectrum utilization.

3) Multi-UAV Coordination and Aerial Computing: Unmanned Aerial Vehicles (UAVs) are critical nodes in 6G non-terrestrial networks. Agentic AI fundamentally alters UAV swarm management from centralized ground control to distributed agent collaboration. For example, the AirVista framework equips UAVs with Multimodal LLMs (MLLMs) acting as autonomous agents, significantly enhancing 3D spatial reasoning for urban logistics and infrastructure monitoring. Concurrently, the UAV-CodeAgents framework leverages the ReAct paradigm to allow UAVs to autonomously plan missions based on satellite imagery and natural language instructions, dynamically adjusting trajectories to maintain line-of-sight communication and avoid obstacles without human telemetry inputs.

C. Real-World Specialized Deployments

1) Collaborative Medical Decision Support and ICU Triage: In safety-critical clinical environments, Agentic AI facilitates distributed medical reasoning. Unlike monolithic diagnostic models, agentic clinical systems utilize role-based delegation. For instance, within a hospital Intensive Care Unit (ICU), a Diagnostic Agent continuously analyzes real-time vitals and lab results to detect early sepsis risk. Concurrently, a History Retrieval Agent parses Electronic Health Records (EHRs) to extract comorbidities, while a Treatment Planning Agent cross-references the patient's state with current clinical guidelines (e.g., Surviving Sepsis Campaign) to propose fluid protocols. A central orchestrator integrates these insights via a shared episodic memory layer, surfacing conflicts for a human physician's review. This distributed intelligence minimizes the diagnostic errors previously observed in early, non-agentic clinical AI (e.g., IBM Watson for Oncology and Epic Sepsis models), which suffered from severe brittleness when facing out-of-distribution real-world data.

2) Scientific Discovery and The AI Scientist: Agentic AI is accelerating the pace of scientific inquiry. Frameworks like the "AI Scientist" automate the entire lifecycle of research. Specialized retriever agents autonomously conduct literature reviews across academic databases, alignment agents identify knowledge gaps, and generative agents propose novel experimental hypotheses. Utilizing code-execution tools, these agents can run simulations, analyze the resulting data, and ultimately employ formatting agents to draft comprehensive, citation-rich research manuscripts.

3) Embodied Intelligence and Vision-Language-Action Robotics: Bridging the digital-physical divide, Agentic AI empowers embodied systems through Vision-Language-Action (VLA) models such as Google's RT-2 and Gemini Robotics. In agricultural deployments, multi-robot harvesting swarms coordinate via shared spatial memory. Mapping drones utilize VLMs to identify diseased fruit clusters and generate high-resolution yield maps, communicating asynchronous spatial coordinates to terrestrial picker robots and transport bots. The central orchestrator dynamically adjusts task allocations in response to mechanical faults or changing weather, achieving a level of autonomous physical synchronization previously impossible with rigid, rule-based robotics.

VI. TRUSTWORTHY AGENTIC AI, SYSTEM SECURITY & ETHICS

The transition from reactive generative models to proactive, goal-directed Agentic AI systems exponentially amplifies the potential threat surface. When AI systems are granted the autonomy to execute code, interface with external APIs, and persistently modify their own memory states, isolated model-level alignment (e.g., RLHF) is insufficient. Trustworthiness in the agentic era demands robust system-level security, rigorous access controls, and comprehensive socio-technical governance.

A. The MAD-BAD-SAD Socio-Technical Framework

To systematically capture the societal and ethical implications of delegated autonomy, researchers have proposed the MAD-BAD-SAD socio-technical framework. This construct posits that the behavior and impact of Agentic AI are co-produced by algorithmic logic, organizational practices, and regulatory environments.

1. **MAD (Motivations, Applications, Dilemmas):** This dimension explores the ethical tensions inherent in autonomous planning. Agentic AI creates a "moral crumple zone"—a phenomenon where accountability for adverse outcomes becomes obfuscated, distributed across the original developers, the deploying organization, and the autonomous agent itself. When an agent makes a dynamic, multi-step plan that results in an unforeseeable negative externality, attributing intent or negligence becomes a profound legal and moral dilemma.
2. **BAD (Biases, Accountability, Dangers):** In multi-agent ecosystems, biases do not merely exist; they compound. Skewed data at the perception stage (e.g., a biased retrieval agent) pollutes the shared memory state, causing downstream reasoning and execution agents to amplify the initial inequity. Furthermore, the lack of human oversight heightens operational dangers. Agentic systems are uniquely vulnerable to data poisoning, where malicious inputs embedded in retrieved documents can silently alter the agent's long-term semantic memory, subverting its future actions without triggering immediate alarms.
3. **SAD (Societal Impact, Adoption Barriers, Design Principles):** The integration of Agentic AI requires restructuring institutional roles and workplace norms. Workforce resistance often emerges not merely from fear of job displacement, but from the psychological friction of being managed or evaluated by non-transparent, autonomous AI orchestrators. Effective adoption necessitates human-centered design principles that maintain meaningful human control, allowing operators to interrupt, audit, and trace the decision-making logic of the agent.

B. Threat Modeling and the Agentic Attack Surface

Agentic AI systems face sophisticated adversarial threats that exploit their defining features: tool usage, memory persistence, and multi-agent communication.

1) The "Lethal Trifecta": Security researcher Simon Willison identifies the "lethal trifecta" of agentic exploitation, which occurs when an agent possesses three simultaneous capabilities: (i) access to private or sensitive data, (ii) the ability to ingest untrusted external content (e.g., reading emails or scraping web pages), and (iii) the authorization to communicate externally (e.g., sending emails or making HTTP requests). When these conditions are met, an attacker can utilize Indirect Prompt Injection—embedding malicious instructions into a seemingly benign webpage or document. When the agent retrieves this document, it unwittingly ingests the attacker's payload as a trusted instruction, potentially causing the agent to package the user's private data and exfiltrate it via an authorized external API call.

2) Zero-Click Attacks and Memory Poisoning: In enterprise deployments (e.g., Microsoft 365 Copilot vulnerabilities like EchoLeak, CVE-2025-32711), indirect prompt injections can be triggered in a "zero-click" manner; the mere act of the agent autonomously summarizing an incoming malicious email can trigger unauthorized data exposure without any explicit user interaction. Furthermore, Memory Poisoning exploits the agent's episodic or semantic memory. By feeding the agent subtly manipulated data over time, an adversary can embed delayed triggers or logical backdoors that steer the agent's future planning modules toward unsafe actions long after the initial injection.

3) Security Post-Mortem Case Studies: OpenClaw and Moltbook: The severity of these vulnerabilities was starkly demonstrated in recent open-source agentic deployments.

- **OpenClaw:** In early 2026, the OpenClaw agent (formerly Clawdbot), which possessed full system shell and file I/O access, suffered catastrophic compromises. Researchers discovered over 900 exposed OpenClaw gateways operating without authentication. Attackers leveraged prompt injection (CVE-2025-6514, CVSS 9.6)

to execute arbitrary remote code and exfiltrate plaintext API keys and OAuth tokens stored in the agent's markdown memory files.

- **Moltbook:** Concurrently, Moltbook—an experimental social network for autonomous AI agents—experienced a massive breach due to a misconfigured Supabase backend. More insidiously, the ecosystem was compromised by malicious "skills" (agent plugins) disguised as benign utilities. An empirical study revealed that 26.1% of analyzed agent skills contained vulnerabilities, with many explicitly designed to scrape private configuration files and secretly transmit credentials to external servers.

C. Defense-in-Depth Mitigation Stack

Securing Agentic AI requires a defense-in-depth assurance stack that permeates the entire Perceive-Plan-Act-Reflect-Learn lifecycle. No single mitigation—such as LLM fine-tuning or input filtering—is sufficient to stop a multi-stage attack.

- **Zero-Trust Intake & Input Sanitization (Perceive):** All external content must be treated as hostile. Agents must utilize strict instruction-hierarchy policies and LLM-as-a-judge filtering to detect prompt injections before the data is written to the agent's context window.
- **Memory Governance (Plan):** To prevent cross-session re-identification and attribute aggregation, memory modules must implement data minimization, context-aware redaction, and strict retention limits (ephemerality).
- **Sandboxing and Least-Privilege Execution (Act):** The execution layer is the most critical vulnerability point. Agents must operate within tightly constrained sandboxes (e.g., Docker containers with restricted capabilities). Tool access must be governed by the principle of least privilege, utilizing ephemeral, scoped credentials managed by a secure vault API rather than plaintext keys stored in memory. Furthermore, Data Loss Prevention (DLP) filters must scan all outbound tool arguments (e.g., outgoing emails or HTTP payloads) to block covert data exfiltration.
- **Tamper-Evident Auditing (Reflect & Learn):** Inter-agent communications must utilize authenticated, encrypted channels. The system must maintain immutable, tamper-evident logs of all retrieved evidence, reasoning rationales, and tool executions to enable post-incident forensics and prevent evaluator spoofing.

Table III: Lifecycle-Aligned Threat Model and Defense Matrix for Agentic AI

Agentic Stage	Primary Threat Vector	Concrete Attack Example	Architectural Mitigation / Control
Perceive	Data Poisoning & Injection	Zero-click Indirect Prompt Injection via untrusted web content.	Input sanitization, Zero-trust intake boundaries, Instruction-data separation.
Plan	Goal Misgeneralization	Memory poisoning leading to persistent unsafe sub-goal formulation.	Constrained Markov Decision Processes (CMDPs), Memory governance & minimization.
Act	Unauthorized Execution	Exfiltration of private credentials via legitimate tool API calls.	Ephemeral credentials, strict sandboxing, Data Loss Prevention (DLP) output filters.

Agentic Stage	Primary Threat Vector	Concrete Attack Example	Architectural Mitigation / Control
Reflect	Evaluator Spoofing	Agent generates plausible but false rationale to hide a policy violation.	Trace auditing, independent formal verification, tamper-evident cryptographic logging.
Learn	Safety Regression	Fine-tuning on corrupted interaction logs strips safety guardrails.	Security regression gating, canary rollouts, strict Supply Chain (SBOM) controls.

VII. EVALUATION METRICS, BENCHMARKING & RELEASE GATING

Evaluating Agentic AI systems introduces foundational challenges that fundamentally diverge from the assessment of static language models or traditional predictive algorithms. While conventional benchmarks evaluate single-turn outputs using localized metrics such as perplexity, BLEU, or classification accuracy, autonomous agents operate along extended temporal trajectories where intermediate decisions, state transitions, tool invocations, and environmental feedback directly govern real-world outcomes. Consequently, evaluating agentic systems requires moving beyond outcome-based accuracy toward rigorous, process-aware methodologies that capture procedural validity, safety boundaries, and systemic trustworthiness.

A. Outcome vs. Process Evaluation

In classical machine learning paradigms, evaluation is predominantly outcome-oriented. In agentic workflows, however, the process itself is the primary vector of both operational utility and vulnerability. An agent may achieve a nominally correct final result while executing unsafe, policy-violating, or ungrounded intermediate actions—such as bypassing database access controls, running unauthorized shell scripts, or leaking sensitive context into external environments. Conversely, an agent might fail to complete a high-level task due to an external environment timeout despite maintaining an entirely sound, verifiable, and safe planning trajectory. Evaluating agents strictly via final task completion obscures these dynamics, demanding a dual evaluation regime that integrates episode-level outcome signals with step-level process metrics.

1) Episode-Level Outcome Metrics:

- **Success Rate (SR):** The proportion of evaluation episodes in which the agent completely satisfies the user's high-level goal without human intervention.
- **Catastrophic Event Rate (CER):** The percentage of trajectories that trigger unacceptable or irreversible real-world damage (e.g., unauthorized mass deletion of production records, severe financial loss, or dangerous physical actuator commands).
- **Preference Head-to-Head Win Rate (PHR):** Pairwise preference evaluation against standard baselines, assessing user satisfaction, coherence, and instruction-following fidelity.

2) Step-Level Process Metrics:

- **Constraint Violation Rate (CVR):** The frequency of hard-rule breaches, security policy violations, or tool-permission exceedances measured across individual steps within an execution trajectory. CVR serves as an immediate diagnostic indicator of whether planning and execution components remain within designated operational envelopes.
- **Decision Coverage Rate (DCR / Trace-Coverage):** The proportion of agent actions that possess complete, auditable execution traces, encompassing explicit reasoning rationales, retrieved evidence citations, invoked tool arguments, and environmental responses. DCR ensures that system behavior is not emergent from uncalibrated stochastic shortcuts.

- **Compliance Verification Rate (CompVR):** The fraction of intermediate actions and outputs that can be formally and independently verified against regulatory standards, formal safety logic, or domain-specific guidelines (e.g., HIPAA compliance in medical workflows, or FINRA market conduct rules in algorithmic finance).
- **Attack Success Rate (ASR) & Refusal Rate (RR):** The vulnerability of the agent to adversarial attacks across its trajectory, specifically measuring whether direct or indirect prompt injections successfully hijack the execution graph, alongside the agent's capacity to correctly refuse unsafe directives.

Reporting step-level violation histograms alongside trajectory-level success metrics prevents the "lucky agent" fallacy, ensuring that autonomous systems released into production are robust against compounding errors and silent failures.

B. The Benchmarking Landscape

Traditional static Question-and-Answer (Q&A) test suites are fundamentally insufficient for assessing agentic capabilities because they lack environmental feedback, state transitions, and long-horizon tool manipulation. To bridge this gap, modern benchmarking frameworks construct sandboxed, interactive virtual environments that emulate real-world systems.

1) Web and Digital Workflow Benchmarks:

- **WebArena:** An enterprise-grade, simulated web environment encompassing dynamic websites (e-commerce storefronts, social platforms, collaborative software repositories, and content management systems). WebArena stresses the agent's capacity to perform multi-hop, long-horizon tasks requiring complex browser interaction. Crucially, empirical evaluations on WebArena demonstrate a substantial human-agent performance gap: while human operators achieve success rates exceeding 78%, frontier LLM agents typically achieve success rates below 15%, exposing fundamental limitations in multi-step visual grounding, state tracking, and recovery from unexpected web navigation errors.
- **OSWorld & OdysseyBench:** Expanding beyond browser environments, OSWorld benchmarks multimodal agents across realistic operating system tasks involving arbitrary GUI interactions, file system modifications, and terminal executions across diverse desktop applications. OdysseyBench extends this to complex, multi-application office workflows (spanning spreadsheets, word processors, PDF parsers, and email clients), systematically measuring long-term memory maintenance and application-switching fidelity.

2) Interactive Emulation and Security Testbeds:

- **ToolEmu:** An LM-emulated sandbox designed to evaluate tool-using safety without real-world side effects. ToolEmu constructs emulators for diverse APIs, allowing evaluators to safely execute high-risk trajectories (e.g., automated financial transfers or administrative privilege modifications) and quantify the precise step at which policy breaks occur.
- **AgentDojo & HarmBench:** Dynamic evaluation environments focused on adversarial resilience. AgentDojo specifically tests the agent's vulnerability to indirect prompt injections embedded in web pages or retrieved databases, evaluating whether layered runtime guardrails can prevent hijacked tool execution. HarmBench standardizes red-teaming methodologies, evaluating attack success across escalating threat models and multi-turn social engineering vectors.

C. Multi-Tier Release Gating Strategy

Given the severe risks associated with autonomous execution, production deployment of Agentic AI systems cannot rely on single aggregate benchmarks. Instead, organizations must implement a risk-stratified, multi-tier release gating pipeline that progressively validates agents through increasing stakes and operational fidelity:

- **Tier 0 (Offline Replay & Deterministic Regression):** The fastest iteration loop. Every software update, prompt alteration, memory policy change, or model weight modification is subjected to deterministic replay against a curated repository of historical incident traces, known prompt injection payloads, and edge-case failure logs. Gating criteria require absolute zero regressions on critical safety scenarios and a CVR equal to zero on all deterministic policy checks.
- **Tier 1 (Simulation Stress-Testing & ODD Profiling):** The agent is evaluated within dynamic, sandboxed environments (such as ToolEmu or domain-specific simulators like CARLA for embodied autonomous navigation). This phase probes performance across defined Operational Design Domains (ODDs), systematically introducing out-of-distribution (OOD) noise, adversarial perturbations, network latencies, and tool failures. Acceptance requires bounding catastrophic risks to near-zero thresholds (e.g., CER < 0.1%) and verifying minimum floors for auditability (e.g., DCR > 95%).
- **Tier 2 (Shadow Mode Deployment):** The agent operates in live environments on real-world input streams, but with its external actuators and write permissions strictly blocked. The agent generates internal reasoning traces, plans, and proposed tool arguments, while human operators execute the actual workflows. This enables the real-time measurement of trace completeness (DCR) and compliance verifiability (CompVR) without exposing the organization to operational harm.
- **Tier 3 (Canary Rollouts & Continuous Production Monitoring):** The agent is granted limited, monitored execution authority over a restricted fraction of production traffic. Continuous telemetry tracks leading risk indicators—such as spikes in constraint violation rates, unusual tool-invocation volumes, permission denials, and evaluator judge disagreements. Automated circuit-breakers trigger immediate execution halts and rollbacks if anomalous behavioral drift is detected, closing the feedback loop into subsequent Tier 0 regression packs.

VIII. OPEN CHALLENGES & FUTURE ROADMAP

While current implementations demonstrate remarkable empirical utility, Agentic AI remains in an early stage of foundational maturity. The path toward highly dependable, general-purpose autonomous intelligence requires resolving systemic architectural, computational, and safety bottlenecks.

A. Lifelong Learning and Continuous Self-Evolution

Contemporary language agents are predominantly static across deployments. While in-context learning and retrieval from episodic memory allow temporary adaptation within a session, agents cannot autonomously synthesize durable procedural knowledge or update internal model parameters without risking catastrophic forgetting or concept drift. When agents interact with evolving real-world environments, changes in APIs, social norms, and domain data gradually erode their performance.

A promising paradigm for overcoming this dependency is demonstrated by self-play and zero-data self-evolution frameworks, such as Absolute Zero Reasoning (AZR). Rather than relying perpetually on static, human-annotated corpora, agents can operate in reinforced self-play environments where they autonomously formulate novel problem tasks, execute candidate solution trajectories, and validate outcomes against deterministic environment checks (e.g., formal compilers or symbolic math kernels). By pairing self-generated tasks with verifiable reward feedback, the agent iteratively bootstraps its own reasoning capabilities without external data contamination. However, integrating continuous parameter updates with long-term memory retention without destabilizing the agent's behavioral alignment remains an open grand challenge.

B. Scalable Oversight and Human-in-the-Loop Verification

As the planning horizons of Agentic AI systems expand from minutes to days or weeks, human-in-the-loop oversight faces a fundamental scalability barrier. A human operator cannot feasibly review hundreds of interconnected tool calls,

intermediate memory retrievals, and distributed agent communications. This creates a critical oversight paradox: if oversight is too granular, it negates the economic and operational advantages of autonomy; if oversight is too sparse, compounding errors and covert misalignments will propagate undetected until catastrophic failure occurs.

Resolving this challenge necessitates the development of hierarchical verification frameworks. Instead of auditing individual primitive actions, human supervisors must be empowered by specialized meta-evaluator agents that audit structural subgoals, summarize decision paths, and highlight anomalies in real time. Furthermore, cognitive architectures must incorporate calibrated metacognition—endowing agents with the explicit self-awareness to recognize uncertainty, flag out-of-distribution inputs, and proactively halt execution to solicit human guidance before performing irreversible actions.

C. Green Agentic AI: Compute Efficiency and Token Economics

The computational and environmental footprint of Agentic AI is scaling unsustainably. While traditional Generative AI incurs inference costs proportional to prompt and completion length, Agentic AI multiplies compute requirements through iterative reasoning loops, multi-agent debates, reflective critique passes, and recursive tree searches. For example, solving a complex coding or mathematical task via multi-agent orchestration or MCTS can require dozens of LLM calls per problem instance, consuming millions of tokens and substantial electrical power.

To achieve sustainable and scalable deployment, the field must advance Green Agentic AI. Key technological frontiers include:

- **Adaptive Test-Time Compute Allocation:** Dynamically modulating the depth of reasoning based on task difficulty, reserving multi-step search (such as Tree of Thoughts) for high-complexity problems while routing routine tasks to fast, single-pass generation.
- **Small Language Model (SLM) Specialization:** Replacing monolithic frontier models with constellations of task-specialized SLMs (e.g., 1B–8B parameter models) fine-tuned for specific roles such as query routing, syntax validation, or citation checking.
- **Context Compression & Efficient KV-Cache Management:** Implementing selective forgetting, hierarchical summarization, and memory pruning to prevent quadratic attention expansion during long-horizon workflows.

D. Neuro-Symbolic Convergence Towards AGI

Perhaps the most profound intellectual vector in the evolution of Agentic AI is the convergence of the neural and symbolic lineages. The neural paradigm provides unprecedented semantic understanding, creative generation, and broad commonsense adaptability, yet it suffers from stochastic opacity, lack of causal understanding, and susceptibility to hallucinations. The symbolic paradigm provides formal verifiability, deterministic logic, and absolute safety boundaries, but collapses when faced with high-dimensional, unstructured real-world ambiguity.

The next generation of autonomous systems must synthesize these complementary approaches into unified neuro-symbolic architectures. In such systems, neural language models function as perceptual encoders, intuitive hypothesis generators, and flexible human interfaces, while symbolic reasoning engines (e.g., PDDL planners, SMT solvers, and formal logic checkers) enforce provable constraints, verify intermediate plans, and execute mathematical deductions. Bridging the statistical flexibility of deep learning with the mathematical rigor of symbolic logic represents the definitive technical foundation for constructing robust, trustworthy, and scalable systems capable of progressing toward Artificial General Intelligence.

IX. CONCLUSION

The emergence of Agentic AI represents a transformative milestone in the trajectory of artificial intelligence, redefining computational systems from passive responders to proactive, autonomous collaborators. As demonstrated throughout this comprehensive survey, agentic capabilities are not an incidental extension of scaling model size; rather, they arise from the architectural synthesis of large-scale neural representations with structured cognitive components—encompassing persistent memory hierarchies, interleaved reasoning-acting frameworks, dynamic tool integration, and multi-agent coordination protocols.

By establishing a dual-paradigm taxonomy, this work has decoupled the historical contributions of symbolic architectures from the stochastic mechanics of neural orchestration, resolving the pervasive challenge of conceptual retrofitting. Empirical case studies across automated software engineering, 6G communication networks, healthcare, and physical robotics confirm that multi-agent systems can achieve remarkable operational efficiencies, yet they simultaneously introduce severe vulnerabilities spanning prompt injection, memory poisoning, coordination failures, and the lethal trifecta of unauthorized tool use.

Addressing these systemic risks demands an equally sophisticated assurance stack: transitioning from outcome-based benchmarking to rigorous process-level evaluation, implementing defense-in-depth security across the agent lifecycle, and establishing multi-tier release gating. Moving forward, realizing the full potential of Agentic AI will depend on our ability to navigate the tension between autonomy and controllability. By embracing neuro-symbolic convergence, green computing economics, and robust socio-technical governance, the scientific community can guide the evolution of Agentic AI toward safe, auditable, and beneficial intelligence that fundamentally augments human capability across the global technological ecosystem.

REFERENCES

- [1] M. R. Morris, J. Sohl-Dickstein, N. Fiedel, T. Warkentin, A. Dafoe, A. Faust, C. Farabet, and S. Legg, "Position: Levels of AGI for operationalizing progress on the path to AGI," in Proc. 41st Int. Conf. Mach. Learn. (ICML), Vienna, Austria, 2024.
- [2] R. Burnell, Y. Yamamori, O. Firat, K. Olszewska, S. Hughes-Fitt, O. Kelly, I. R. Galatzer-Levy, M. R. Morris, A. Dafoe, A. M. Snyder, N. D. Goodman, M. Botvinick, and S. Legg, "Measuring progress toward AGI: A cognitive framework," arXiv preprint arXiv:2605.28405, 2026.
- [3] H. Derouiche, H. Mezni, and Z. Brahmi, "Agentic AI frameworks: Architectures, protocols, and design challenges," arXiv preprint arXiv:2508.10146, 2025.
- [4] J. Qi, M. Li, J. Liu, Y. Shu, D. Yu, S. Ma, W. Cui, Y. Zhao, Y. Chen, R. Jiang, I. King, and Z. Xu, "Towards trustworthy agentic AI: A comprehensive survey of safety, robustness, privacy, and system security," Academia AI and Applications, vol. 2, 2026.
- [5] S. T. Erukude, S. R. Veluru, and V. C. Marella, "Agentic AI - The rise of autonomous intelligent agents in the era of LLMs," Indian J. Comput. Sci. Eng. (IJCSE), vol. 16, no. 1, pp. 9–16, 2025.
- [6] R. Sapkota, K. I. Roumeliotis, and M. Karkee, "Vibe coding vs. agentic coding: Fundamentals and practical implications of agentic AI," arXiv preprint arXiv:2505.19443, 2025.
- [7] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in Proc. 11th Int. Conf. Learn. Represent. (ICLR), Kigali, Rwanda, 2023.
- [8] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," arXiv preprint arXiv:2308.08155, 2023.
- [9] P. K. Donta, A. Saleh, Y. Li, S. Vaishnav, K. Fang, H. Feng, Y. Xia, T. R. Gadekallu, Q. Zhang, X. Shi, A. Beikmohammadi, S. Magnússon, I. Murturi, C. K. Dehury, M. Paprzycki, L. Lovén, S. Tarkoma, and S. Dustdar, "Socio-technical aspects of agentic AI," arXiv preprint arXiv:2601.06064, 2025.
- [10] X. Meng, P. Gao, Z. Ma, and C. Peng, "LLM-based agents for automated bug fixing: How far are we?," arXiv preprint arXiv:2411.10213, 2024.

- [11] M. Abou Ali, F. Dornaika, and J. Charafeddine, "Agentic AI: A comprehensive survey of architectures, applications, and future directions," *Artif. Intell. Rev.*, vol. 59, no. 1, Art. no. 11, 2026.
- [12] P. Peykani, S. Ghanidel, I. Javadi-Sisi, V. Snasel, and S. Mirjalili, "A holistic review of agentic AI frameworks, applications, and research trajectories," *Arch. Comput. Methods Eng.*, 2026.
- [13] J. Schneider, "Generative to agentic AI: Survey, conceptualization, and challenges," *arXiv preprint arXiv:2504.18875*, 2025.
- [14] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, 1995.
- [15] R. Sapkota, K. I. Roumeliotis, and M. Karkee, "AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges," *Inf. Fusion*, vol. 126, Art. no. 103599, 2026.
- [16] F. Jiang, C. Pan, L. Dong, K. Wang, O. A. Dobre, and M. Debbah, "From large AI models to agentic AI: A tutorial on future intelligent communications," *arXiv preprint arXiv:2505.22311*, 2025.
- [17] T. R. Sumers, S. Yao, K. Narasimhan, and T. L. Griffiths, "Cognitive architectures for language agents," *Trans. Mach. Learn. Res. (TMLR)*, vol. 2024, no. 2, pp. 1–32, 2024.
- [18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.
- [19] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 1877–1901.
- [20] 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.
- [21] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al., "DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.
- [22] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023, pp. 11809–11822.
- [23] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk, et al., "Graph of thoughts: Solving elaborate problems with large language models," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 16, 2024, pp. 17682–17690.
- [24] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 9459–9474.
- [25] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proc. 36th Annu. ACM Symp. User Interface Softw. Technol. (UIST)*, 2023, pp. 1–22.
- [26] C. Packer, V. Fang, S. G. Patil, K. Lin, S. Wooders, and J. E. Gonzalez, "MemGPT: Towards LLMs as operating systems," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024.
- [27] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023, pp. 68539–68551.
- [28] 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," *Trans. Mach. Learn. Res. (TMLR)*, 2023.
- [29] S. Hong, X. Zheng, J. Chen, Y. Cheng, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, et al., "MetaGPT: Meta programming for multi-agent collaborative framework," *arXiv preprint arXiv:2308.00352*, 2023.
- [30] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan, "SWE-bench: Can language models resolve real-world GitHub issues?," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024.
- [31] J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, and O. Press, "SWE-agent: Agent-computer interfaces enable automated software engineering," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, 2024, pp. 50528–50652.
- [32] C. S. Xia, Y. Deng, S. Dunn, and L. Zhang, "Agentless: Demystifying LLM-based software engineering agents," *arXiv preprint arXiv:2407.01489*, 2024.
- [33] S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, S. Yuan, X. Cheng, Y. Bisk, D. Fried, et al., "WebArena: A realistic web environment for building autonomous agents," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024.

- [34] T. Xie, D. Zhang, J. Chen, X. Li, S. Zhao, R. Cao, J. Zhou, B. Wang, Z. Chen, T. Yu, et al., "OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, 2024, pp. 52040–52094.
- [35] Y. Ruan, H. Dong, A. Wang, S. Pitis, Y. Zhou, J. Ba, and J. Yang, "Identifying the risks of LM agents with an LM-emulated sandbox," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024.
- [36] E. Debenedetti, J. Zhang, M. Balunovic, L. Beurer-Kellner, M. Fischer, and F. Tramèr, "AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, 2024, pp. 82895–82920.
- [37] M. Mazeika, L. Phan, X. Yin, A. Zou, Z. Wang, N. Mu, E. Sakhaee, N. Li, S. Basart, B. Li, et al., "HarmBench: A standardized evaluation framework for automated red teaming and robust refusal," in *Proc. 41st Int. Conf. Mach. Learn. (ICML)*, 2024.
- [38] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in *Proc. 16th ACM Workshop Artif. Intell. Secur. (AISec)*, 2023, pp. 79–90.
- [39] A. Zhao, Y. Wu, Y. Yue, T. Wu, Q. Xu, M. Lin, S. Wang, Q. Wu, Z. Zheng, and G. Huang, "Absolute Zero: Reinforced self-play reasoning with zero data," *arXiv preprint arXiv:2505.03335*, 2025.
- [40] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al., "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Proc. Conf. Robot Learn. (CoRL)*, 2023, pp. 2165–2183.
- [41] C. Lu, C. Lu, R. T. Lange, J. Foerster, J. Clune, and D. Ha, "The AI Scientist: Towards fully automated open-ended scientific discovery," *arXiv preprint arXiv:2408.06292*, 2024.
- [42] A. Ehtesham, A. Singh, G. K. Gupta, and S. Kumar, "A survey of agent interoperability protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP)," *arXiv preprint arXiv:2505.02279*, 2025.
- [43] S. Marro, E. La Malfa, J. Wright, G. Li, N. Shadbolt, M. Wooldridge, and P. Torr, "A scalable communication protocol for networks of large language models," *arXiv preprint arXiv:2410.11905*, 2024.
- [44] S. Willison, "The lethal trifecta for AI agents: Private data, untrusted content, and external communication," *Simon Willison's Weblog*, Jun. 2025.
- [45] Noma Security Research, "Clawdbot security crisis: Open-source AI agent exposes critical vulnerabilities in agentic AI deployments," *Noma Security Technical Report*, Jan. 2026.
- [46] J. O'Reilly, "Moltbook database breach: AI agent social network exposes API keys for 32,000 agents," *404 Media Report*, Jan. 2026.
- [47] J. E. Laird, *The Soar Cognitive Architecture*. Cambridge, MA, USA: MIT Press, 2012.
- [48] A. S. Rao and M. P. Georgeff, "BDI agents: From theory to practice," in *Proc. 1st Int. Conf. Multi-Agent Syst. (ICMAS)*, San Francisco, CA, USA, 1995, pp. 312–319.
- [49] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
- [50] National Institute of Standards and Technology (NIST), "Artificial intelligence risk management framework (AI RMF 1.0)," *NIST AI 100-1*, Jan. 2023.