Agentic Services Computing

Shuiguang Deng, *Senior Member, IEEE,* Hailiang Zhao, *Member, IEEE,* Ziqi Wang, *Student Member, IEEE,* Guanjie Cheng, Peng Chen, Wenzhuo Qian, Zhiwei Ling, Jianwei Yin, Albert Y. Zomaya, *Fellow, IEEE,* Schahram Dustdar, *Fellow, IEEE*

***Abstract*—The rise of large language model (LLM)-powered agents is transforming services computing, moving it beyond static, request-driven functions toward dynamic, goal-oriented, and socially embedded multi-agent ecosystems. We propose Agentic Services Computing (ASC), a paradigm that reimagines services as autonomous, adaptive, and collaborative agents capable of perceiving, reasoning, acting, and evolving in open and uncertain environments. We organize ASC around a four-phase lifecycle: Design, Deployment, Operation, and Evolution. It is examined through four interwoven research dimensions: (i) perception and context modeling, (ii) autonomous decision-making, (iii) multi-agent collaboration, and (iv) evaluation with alignment and trustworthiness. Rather than functioning as isolated layers, these dimensions evolve together. Contextual grounding supports robust deployment; autonomous reasoning drives real-time action; collaboration emerges from agent interaction; and trustworthiness is maintained as a lifelong, cross-cutting commitment across all lifecycle stages. In developing this framework, we also survey a broad spectrum of representative works that instantiate these ideas across academia and industry, mapping key advances to each phase and dimension of ASC. By integrating foundational principles of services computing with cutting-edge advances in LLM-based agency, ASC offers a unified and forward-looking foundation for building intelligent, accountable, and human-centered service ecosystems.**

***Index Terms*—Agentic services computing, LLM agents, multi-agent systems, service lifecycle.**

I. Introduction

The field of services computing is undergoing a paradigm shift that fundamentally redefines what constitutes a "service". Traditionally, services have been modeled as function-oriented, modular components, typically exposed as APIs or microservices, designed for predictable request-response interactions. The advent of large language models (LLMs) and multimodal foundation models has catalyzed the emergence of a new class of software entities: *agents*. These agents are autonomous, goal-driven, and context-aware; they can perceive environments, reason over tasks, execute actions, and collaborate with users and other agents. Building on this transformation, we propose *Agentic Services Computing* (ASC), a paradigm in which services are no longer invoked but orchestrated, no longer reactive but proactive, and no longer static but evolving participants in dynamic ecosystems.

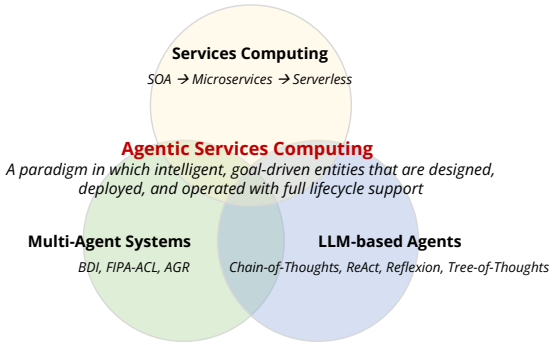

Figure 1: We define Agentic Services Computing as an emergent paradigm that reimagines services as adaptive, goal-driven participants in a dynamic ecosystem, formed through the convergence of services computing, multi-agent systems, and LLM-based agents.

Historically, the integration of MAS and services computing relied on symbolic, rule-based architectures. The Belief-Desire-Intention (BDI) model [38], [76] provided a cognitive architecture in which *Beliefs* represent an agent's knowledge of the world, *Desires* encode its goals, and *Intentions* denote actions to which the agent has committed. This architecture was implemented in practical systems such as PRS [77] and dMARS [59]. To facilitate interoperability, FIPA standards [71] introduced a standardized Agent Communication Language (ACL) along with protocols such as Contract Net for service discovery [99], composition [197], [128], and negotiation [137], [186]. Platforms including JADE [32], Jason [37], and SPADE [169] operationalized these principles and enabled the development of distributed, cooperative agent systems. However, these approaches were constrained by hand-crafted knowledge bases and rigid planning mechanisms, limiting their adaptability in open, dynamic environments.

The rise of LLMs and vision-language models (VLMs) [34] has overcome many of these limitations. These models serve as general-purpose cognitive engines that enable agents to interpret natural language, process multimodal input, and generate contextually appropriate actions with minimal task-specific programming. The ReAct paradigm [195] exemplifies this transformation by interleaving *Thought* and *Action* within a closed reasoning-acting loop. Systems such as Voyager [177] extend this capability by supporting autonomous exploration and self-improvement in complex domains. Benchmarking frameworks like WebArena [204] and VisualWebArena [105] evaluate agent

Shuiguang Deng and Hailiang Zhao are corresponding authors.

Shuiguang Deng, Peng Chen, Wenzhuo Qian, and Jianwei Yin are with College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mails: {dengsg, pgchen, qwz, zjuyjw}@zju.edu.cn).

Hailiang Zhao, Ziqi Wang, Guanjie Cheng, and Zhiwei Ling are with School of Software Technology, Zhejiang University, Ningbo 315048, China (e-mails: {hliangzhao, wangziqi0312, chengguanjie, zwling}@zju.edu.cn).

Albert Y. Zomaya is with School of Computer Science, University of Sydney, Sydney, NSW 2006, Australia (e-mail: albert.zomaya@sydney.edu.au).

Schahram Dustdar is with the Distributed Systems Group, TU Wien, Vienna, Austria, and also with ICREA, Universitat Pompeu Fabra, Barcelona, Spain (e-mail: dustdar@dsg.tuwien.ac.at).

Table I: A comparative analysis of survey perspectives on agentic services. Our work uniquely integrates historical foundations, modern LLM capabilities, and a full service lifecycle.

Work	Governance & Trust [198], [70], [95]	Architectures & Collaboration [179], [107]	Domain-Specific Applications [196], [64], [193], [171], [115], [88]	Classical MAS [63]	Agent Evolution [180]	Ours
Covers Classical MAS	✗	✗	✗	✓	✓	✓
Covers LLM Agents	✓	✓	✓	✗	✓	✓
Multimodal Integration	Implied	Text-Centric	Domain-Dependent	Limited	✗	**Explicit**
E2E Service Lifecycle	✗	✗	Partial	✗	✗	✓
Goal-Driven Composition	✗	✗	✗	✗	✗	✓
Unified Governance Model	Partial	✗	✗	✗	✗	✓

performance on real-world tasks. At the system level, platforms such as AutoGen [188] and Magentic-One [72] facilitate collaboration among teams of specialized agents executing intricate workflows. This evolution marks a revival of coordinated intelligence, now powered by neural cognition rather than symbolic reasoning.

Yet as agents become increasingly autonomous, they introduce new challenges in governance, safety, and trust. The interaction model has fundamentally shifted: from static, protocol-driven exchanges to dynamic, cognition-driven engagements that continuously evolve over time. In this new landscape, a service is no longer a fixed function awaiting invocation; it is a proactive entity capable of perception, reasoning, action, and collaboration. This transformation exposes critical limitations in existing paradigms when considered in isolation. Classical MAS offer rich theoretical models of autonomy and coordination, but lack systematic support for full-lifecycle engineering, operational governance, and integration into real-world service infrastructures. Traditional services computing provides robust principles for designing, deploying, and managing distributed systems [197], [128], yet it treats services as passive, stateless components devoid of cognitive capabilities or behavioral agency. Meanwhile, LLM-based agents exhibit remarkable reasoning and adaptability [195], [177], but often lack mechanisms for auditability, value alignment [198], safety [95], and long-term operational rigor. ASC addresses these gaps not by combining the three domains additively, but by unifying them into a coherent discipline. As illustrated in Figure 1, ASC integrates cognitive autonomy, multi-agent coordination, and lifecycle-centric service engineering to support the development and governance of intelligent services that are both capable and trustworthy. Within this paradigm, *agentic services* are first-class citizens whose identity is defined not by static interfaces, but by their intent, behavior, and evolutionary trajectory. In our opinion, ASC reorients the focus of service design: from service invocation to agent orchestration, from predefined workflows to emergent collaboration, and from functional correctness to behavioral trustworthiness. ASC embeds technical capabilities within a structured agentic service lifecycle, ensuring continuity, resilience, and adaptability across deployment, operation, and evolution.

While numerous surveys on MAS and LLM agents exist, they

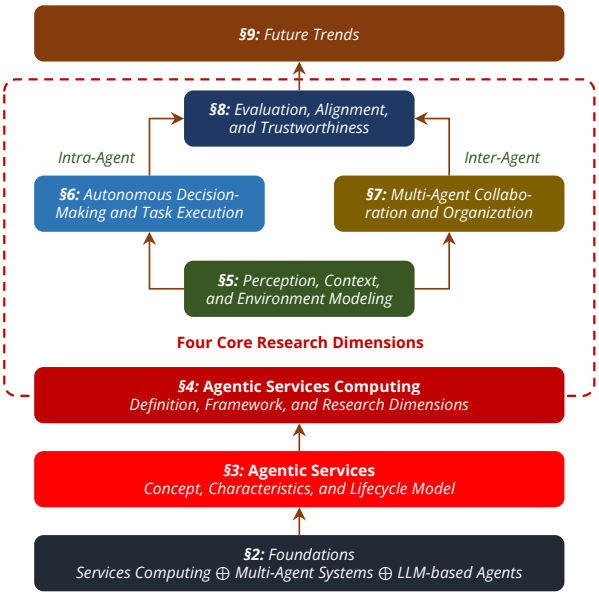

Figure 2: The orchestration of core sections in this paper.

exhibit critical limitations. As shown in Table I, current works fall into disconnected categories. Some provide taxonomies of agent architectures [179], [107] but lack historical context and comprehensive coverage of the service lifecycle. Domain-specific surveys in robotics [196], medicine [171], and software engineering [115], [88] offer deep insights but remain isolated from classical MAS and service engineering principles. Surveys on governance [198], [70], [95] treat trust and safety in isolation. Classical MAS surveys [63] do not incorporate the capabilities enabled by LLMs. Although [180] bridges classical and modern paradigms, it omits engineering considerations and lifecycle management. To the best of our knowledge, no existing survey integrates the three foundational pillars: classical MAS principles, LLM & VLM cognitive capabilities, and lifecycle-centric services computing. Our core contribution is a unified framework for ASC that bridges these domains. This framework, detailed in Section IV, establishes ASC as a principal discipline by defining core concepts, identifying essential features, and proposing a structured research agenda. A central innovation is the agentic service lifecycle as the organizing principle, ensur-

ing that technical capabilities are analyzed within the context of service continuity, operational resilience, and evolutionary adaptability.

The remainder of this paper is structured as shown in Figure 2. Section II traces the evolution from classical MAS to modern LLM-based agents. Section III introduces agentic services and a four-phase lifecycle model. Section IV formalizes the ASC paradigm with a structured research framework. Sections V through VIII examine technical foundations through the lifecycle lens. Section V covers perception and environment modeling. Section VI investigates decision-making and execution. Section VII explores multi-agent collaboration. Section VIII focuses on evaluation, value alignment, and trustworthiness. Section IX discusses trends and challenges. Section X concludes.

II. Foundations

ASC emerges from the convergence of three foundational paradigms: services computing, MAS, and LLM-based agents. Each paradigm contributes distinct and essential capabilities. Namely, engineering rigor and lifecycle governance from services computing; decentralized autonomy, coordination, and social interaction from MAS; and generative cognition with natural language understanding from LLM-based agents. While each domain has significantly advanced the design of intelligent systems, all face inherent limitations when applied in isolation, particularly in open and dynamic environments. This section examines their complementary strengths and shared challenges, arguing that only through deep, principled integration can their collective potential be fully realized. ASC synthesizes these paradigms into a coherent framework that inherits their core advantages while mitigating their individual shortcomings, thereby enabling the development of robust, adaptive, and intelligent service ecosystems.

A. Services Computing

Services computing provides the engineering backbone for constructing modular, scalable, and maintainable distributed service systems. Its evolution spans from service-oriented architecture (SOA) [133], [36], [134], [57], [199] to microservices [127] and, more recently, serverless computing [29]. This progression has steadily enhanced system agility, scalability, and operational efficiency through increasingly fine-grained decomposition of functionality. At the heart of this paradigm lies the principle of encapsulating functionality as reusable, independently managed services that expose well-defined interfaces. SOA established standardized protocols such as WSDL, SOAP, and REST, along with orchestration languages like BPEL [54], to ensure interoperability across heterogeneous platforms. Microservices extended this vision by decomposing monolithic applications into independently deployable units, thereby supporting continuous integration and delivery practices. Serverless computing further abstracts infrastructure concerns by offering event-driven execution models with automatic scaling and pay-per-use pricing, as exemplified by platforms such as AWS Lambda and Azure Functions. A defining characteristic of services computing is its emphasis on end-to-end lifecycle governance. The ISO/IEC/IEEE 15288:2023 standard [97] prescribes systematic processes for the conception, development, operation, and retirement of systems, encompassing configuration management, risk assessment, and verification activities. Operational practices reinforce this governance through mechanisms such as service-level agreements (SLAs), monitoring tools (e.g., Prometheus [8]), visualization dashboards (e.g., Grafana [4]), and distributed tracing frameworks (e.g., OpenTelemetry [7]), all of which ensure observability, reliability, and regulatory compliance. Additionally, frameworks like ITIL [73] institutionalize best practices for service management across organizational contexts.

Notwithstanding these strengths, traditional service models remain function-centric and statically defined. They lack intrinsic autonomy, goal-directed behavior, and the capacity for adaptive reasoning. Consequently, they perform effectively within deterministic, pre-specified workflows but are fundamentally unable to support entities that must perceive their environment, formulate plans, learn from experience, or evolve in response to changing conditions.

B. Multi-Agent Systems

MAS model decentralized intelligence through autonomous entities that perceive their environments, reason about goals, and coordinate with one another to achieve complex objectives [185]. These agents exhibit both proactivity and reactivity, enabling them to act independently and adapt to dynamic conditions without requiring explicit external control. A foundational cognitive architecture for such agents is the BDI model [76], which structures decision-making around three internal mental states. In this framework, *Beliefs* represent an agent's knowledge of the environment, *Desires* encode its objectives, and *Intentions* correspond to the courses of action to which the agent has committed. To support interoperability and structured interaction, the Foundation for Intelligent Physical Agents (FIPA) established formal standards [71] that define message formats and interaction protocols. Among these, the Contract Net protocol [158] is widely used for task negotiation and service allocation. Practical implementations of these theoretical constructs include systems such as PRS [77], dMARS [59], and Jason [37], which operationalize BDI reasoning and agent communication in real-world settings. To balance deliberation and responsiveness, hybrid agent architectures integrate high-level planning with low-level reactive mechanisms. An influential example is the subsumption architecture [41], which enables layered behavior control without centralized reasoning. Furthermore, organizational models such as AGR (Agent, Group, Role) [69] provide abstractions for structuring large-scale multi-agent systems, thereby facilitating scalable coordination and role-based interaction.

Despite these advances, classical MAS faces limitations in modern service contexts. Its reliance on hand-engineered rules and domain-specific knowledge restricts adaptability in open and evolving environments. As system scale increases, communication overhead grows nontrivially, and the verification of emergent collective behaviors becomes increasingly difficult. Most critically, classical MAS lacks native integration with contemporary service infrastructure, e.g., versioning systems, con-

tinuous deployment pipelines, runtime monitoring, and service-level agreement (SLA) enforcement mechanisms. This gap has impeded the adoption of MAS principles in production-grade, industrial-scale service ecosystems.

C. LLM-based Agents

LLM-based agents utilize LLMs to generate human-like language and carry out complex reasoning tasks. The Transformer architecture [175] laid the foundation for these capabilities by introducing self-attention mechanisms, which enable efficient parallelization and the modeling of long-range dependencies in sequential data. This architectural innovation gave rise to increasingly powerful models, including GPT [141], GPT-2 [142], GPT-3 [42], and GPT-4 [23]. These models demonstrate emergent abilities in few-shot learning, multi-step reasoning, and cross-domain generalization, even without explicit task-specific training. These LLMs serve as cognitive engines for agentic systems, which are instantiated through several distinct architectural patterns. Reactive Agents rely on prompting strategies such as Chain-of-Thought [183] to perform stepwise reasoning; however, they lack persistent memory and the capacity for long-term planning. In contrast, Act-and-Reason Agents interleave reasoning with action execution. A representative example is REACT [195], which generates explicit thought traces while invoking external tools, such as web search or code interpreters, to interact with dynamic environments. A third category, Reflective Agents, incorporates mechanisms for self-evaluation and iterative improvement. For instance, Reflexion [156] simulates internal reflection to refine future decisions, and Tree-of-Thoughts [194] explores multiple reasoning trajectories to identify globally optimal solutions. Frameworks such as AutoGPT [192] and BabyAGI [2] illustrate how these agents can autonomously pursue high-level goals through recursive task decomposition and execution. Meanwhile, evaluation platforms like AgentBench [116] provide standardized benchmarks for assessing agent performance across diverse domains. Collectively, these systems exhibit unprecedented levels of adaptability and natural language fluency.

Nevertheless, LLM-based agents are frequently developed as isolated cognitive prototypes, disconnected from operational service infrastructures. They also lack essential engineering features such as version control, runtime monitoring, SLA enforcement, and systematic failure recovery mechanisms. Moreover, their black-box nature introduces significant challenges related to reliability, safety, auditability, and reproducibility. Without a formal engineering discipline that governs their design, deployment, and evolution, their use in mission-critical applications remains fragile and impractical.

D. Convergence: Toward Agentic Services Computing

The limitations inherent in each foundational domain highlight a compelling need for their integration. Services computing provides strong engineering rigor and lifecycle management but lacks intrinsic intelligence and autonomous behavior. Multi-agent systems offer models of autonomy and coordination yet struggle with scalability and integration into modern operational infrastructure. LLM-based agents deliver remarkable cognitive flexibility and natural language fluency but remain largely disconnected from systematic engineering practices, resulting in insufficient reliability and poor system-level integration.

ASC addresses these gaps by unifying the three paradigms into a coherent, next-generation computing model. ASC defines agentic services as autonomous, goal-driven entities that integrate generative cognition, decentralized coordination, and full-lifecycle engineering discipline. These entities are not merely APIs wrapped with large language models; rather, they function as intelligent participants within dynamic service ecosystems. Specifically, agentic services operate under SLAs, are observable through standard monitoring and tracing tooling, and possess the capacity to perceive their environment, plan actions, adapt to changing conditions, and collaborate with other agents and users through natural language. ASC introduces three foundational shifts in how intelligent services are conceived and deployed. First, it moves from state-based execution to goal-driven execution. Second, it transitions from passive service invocation to active, intent-aware engagement. Third, it replaces static orchestration with dynamic, decentralized coordination. Together, these shifts enable the scalable deployment of intelligent agents in real-world applications, including customer service automation, advanced digital assistants, and autonomous operational systems.

III. AGENTIC SERVICE: CONCEPT, CHARACTERISTICS, AND LIFECYCLE MODEL

A. The Structured Characterization of Agentic Services

At the core of ASC lies a fundamental reimagining of the service as a computational entity. In our definition, agentic services are persistent, goal-oriented agents that can initiate actions, maintain internal state, reason about their environments, and adapt autonomously. Rather than functioning as passive endpoints, they act as active participants in dynamic and open-ended ecosystems, thereby fundamentally transforming how distributed service systems are designed, operated, and governed. This paradigm shift is formalized in the SCALE framework, which articulates five interrelated characteristics that collectively define agentic services. As illustrated in Figure 3, agentic services diverge from classical services across multiple dimensions: they replace reactive invocation with proactive agency, isolated transactions with contextual continuity, and static composition with emergent collaboration.

a) Stateful and Context-Aware Interaction: Traditional services process each request in isolation and discard contextual information after the interaction concludes. In contrast, agentic services maintain persistent memory of user interactions, environmental states, and past decisions, which enables coherent, personalized, and long-horizon engagement across multiple sessions. This memory is typically implemented through vector databases such as Pinecone [19] and Weaviate [21], knowledge graphs, or session-aware storage layers. Nevertheless, persistent memory introduces significant challenges, including the need to ensure data privacy and regulatory compliance (e.g., GDPR [173]), maintain consistency across distributed agents, and prevent cognitive overload caused by excessive context accumulation. Consequently, effective memory management re-

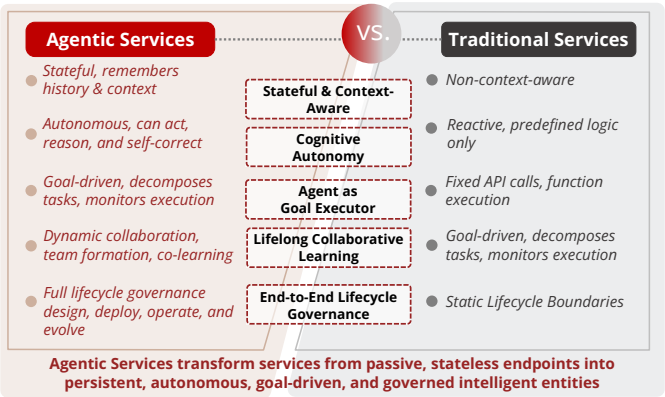

Figure 3: Comparison between agentic and traditional services.

quires mechanisms for selective retention, contextual summarization, and fine-grained access control.

b) Cognitive Autonomy: Whereas classical services operate in a purely reactive manner, agentic services exhibit cognitive autonomy by proactively managing internal states, planning sequences of actions, and adapting strategies based on experience. This capability is supported by LLM-based reasoning architectures such as REACT [195] and Reflexion [156], which enable stepwise reasoning, self-correction, and reflective learning. Coupled with persistent runtimes, these architectures shift the service model from ephemeral function calls to continuous agency. However, such autonomy entails risks, including unbounded resource consumption, execution of unauthorized actions, and difficulties in debugging non-deterministic behaviors. Therefore, autonomy must be constrained by safety guards, policy-based limits, and comprehensive audit trails.

c) Agent as Goal Executor: Traditional services are invoked through fixed interfaces such as REST APIs to execute predefined functions. Agentic services, by contrast, respond to high-level goals. For instance, "Prepare a market analysis report for product *X*." Upon receiving such a goal, the agent decomposes it into subtasks, selects appropriate tools (e.g., web search, code execution, or external API calls), sequences actions, and monitors progress toward successful completion. This shift from specifying *how* a task should be performed to declaring *what* outcome is desired enables flexible and adaptive behavior in unpredictable environments. Yet it also introduces new challenges, including ambiguity in goal interpretation, fidelity of execution, and accountability for emergent or unintended outcomes. To address these issues, formal goal specification languages and robust outcome validation mechanisms are essential for reliable deployment.

d) Lifelong Collaborative Learning: Conventional service composition relies on predefined workflows and static orchestration. Agentic services, however, engage in emergent collaboration by dynamically forming coalitions, negotiating roles, delegating tasks, and resolving conflicts through established protocols such as the Contract Net [158] or auction-based mechanisms [154]. This self-organizing coordination enhances system resilience and collective intelligence. Realizing such collaboration demands advances in decentralized control, incentive alignment, and stability analysis within evolving multi-agent ecosystems. Furthermore, agents can learn from past interactions, e.g., by refining coordination strategies or optimizing team composition, thereby enabling lifelong, collective adaptation that improves performance over time.

e) End-to-end Lifecycle Governance: As autonomy and persistence increase, so does the necessity for comprehensive governance throughout the agent's entire lifecycle. End-to-end lifecycle governance ensures accountability, transparency, and trust from inception to decommissioning. It encompasses three key aspects:

- Versioning of agent logic, memory schemas, policies, and training data;
- Auditing for safety, fairness, regulatory compliance, and ethical alignment [100];
- Secure decommissioning procedures to prevent "zombie agents" from operating unattended.

Drawing upon principles from DevSecOps [31], AI ethics [100], and GDPR [173], this governance model ensures that scalability does not come at the cost of responsibility.

Collectively, the SCALE framework provides a principled foundation that harmonizes cognitive intelligence with engineering discipline, thereby enabling the systematic design, deployment, and governance of agentic services in real-world service ecosystems.

B. The Lifecycle Model of Agentic Services

We formalize the lifecycle of an agentic service as a continuous, closed-loop process comprising four phases: DESIGN, DEPLOYMENT, OPERATION, and EVOLUTION. This model extends traditional paradigms such as SOA [134], microservices [127], and DevOps [31] by integrating cognitive reliability, adaptive coordination, and ethically grounded autonomy. Unlike static services, agentic services are engineered for persistence and continuous evolution, wherein operational feedback directly informs subsequent adaptation and redesign.

a) Design: The DESIGN phase establishes the cognitive and structural blueprint of agentic services. It builds upon classical MAS methodologies, including Gaia [187] and Tropos [39], with an emphasis on agent roles, goals, and social dependencies. Using the AGR framework [69], designers specify organizational topologies, e.g., hierarchical or peer-to-peer structures, and coordination protocols, including the Contract Net [158] and auction-based mechanisms [154]. Cognitive workflows are modeled using LLM-based agent architectures such as REACT [195], Reflexion [156], and Tree-of-Thoughts [194], and are formalized into modular components: perception, planning, action, and reflection. These components are orchestrated through workflow engines like LangGraph [14] or Cadence [18]. Design also embeds safety-by-design principles, incorporating content moderation filters [56], action guards [178], and fallback mechanisms [85] to mitigate operational risks. To ensure interoperability with legacy systems, agents expose standardized interfaces such as REST or gRPC, along with service contracts that align with SOA principles [134]. Domain-specific languages (DSLs) or UML extensions for agent systems [44] further support early validation through simulation and model checking [50].

b) Deployment: The DEPLOYMENT phase instantiates agentic services on scalable and resilient infrastructure. Containerization via Docker [121] and orchestration through Kubernetes [43] enable dynamic, large-scale lifecycle management. For event-driven or sporadic workloads, serverless platforms such as AWS Lambda and OpenFaaS [109] provide cost-efficient, auto-scaling execution environments well suited to goal-driven agents. Continuous integration and continuous delivery (CI/CD) pipelines [31] automate testing, security scanning, and configuration validation to ensure reliable deployment. Agent-specific artifacts, including prompts, tool bindings, policies, and memory schemas, are versioned using MLOps tools such as MLflow [126], Weights&Biases [35], or DVC [108], thereby guaranteeing reproducibility and auditability. Deployment strategies such as canary releases [182], blue-green deployment [191], and feature flags [66] facilitate gradual rollouts and rapid rollbacks. Service meshes like Istio [98] and Linkerd [53] enhance reliability through secure communication, traffic shaping, circuit breaking, and retry mechanisms. In edge computing environments, efficient deployment requires optimized image distribution. PeerSync [58], a decentralized peer-to-peer system for edge computing, reduces cross-network traffic through network-aware downloading, autonomous tracker election, and dynamic caching, thereby accelerating container startup and improving responsiveness under resource constraints.

c) Operation: The OPERATION phase ensures reliable, secure, and accountable execution in production. Given the non-deterministic nature of LLM-based reasoning, operational practices extend beyond conventional monitoring to encompass *cognitive observability* [60], which captures not only system-level metrics but also internal reasoning traces and decision pathways. Key operational practices include the following:

- *Structured logging and tracing:* OpenTelemetry [7] records structured execution traces that include reasoning steps, tool invocations, and inter-agent messages. Distributed tracing systems such as Jaeger [52] provide end-to-end visibility across complex multi-agent workflows.
- *Metrics and alerting:* Core performance indicators, including latency, error rates, token consumption, and cost, are monitored using Prometheus [8] and visualized via Grafana [4]. Service-level objectives (SLOs) guide incident response and capacity planning in accordance with Google's Site Reliability Engineering (SRE) model [33].
- *Anomaly detection:* Statistical methods and LLM-based validators identify hallucinations, policy violations, or infinite loops [48], enabling proactive intervention before failures propagate.
- *Human-in-the-loop (HITL):* For high-stakes or ambiguous decisions, frameworks such as Argilla [27] integrate real-time human feedback, corrections, and learning signals to refine agent behavior iteratively.

Security adheres to zero-trust principles [160], implementing identity-based access control (e.g., OAuth2, OpenID Connect) and encryption both in transit and at rest. Regulatory compliance, such as with GDPR [173], is enforced through comprehensive audit logs and data provenance tracking. Efficient task scheduling and resource allocation are critical to operational efficiency. OBTA [201] leverages data locality and job reordering to minimize task completion time. ONSOCMAX [202] is an online scheduling framework that maximizes social welfare by utilizing a spatiotemporal resource pool and a marginal cost-based pseudo welfare function, achieving a provable α-competitive ratio ($\alpha > 2$) without assumptions about request distributions.

d) Evolution: The EVOLUTION phase enables continuous adaptation, transforming static deployments into self-improving systems. It closes the lifecycle loop by feeding operational insights, e.g., failure patterns, performance bottlenecks, and user feedback, back into the DESIGN phase. This phase integrates principles from MLOps [106] and continual learning [135], adapted to the unique characteristics of LLM-based agents:

- *Online adaptation:* Agents employ self-reflection mechanisms such as Reflexion [156] to analyze past actions and refine future decisions at runtime. Memory-augmented reasoning and in-context learning [124], [61] support real-time behavioral adjustment without full model retraining, thereby enabling low-latency responsiveness.
- *Offline learning:* Historical execution logs are used for deeper model refinement, including prompt optimization, fine-tuning of base LLMs, or retraining of policy networks. Alignment techniques such as reinforcement learning from human feedback (RLHF) [48] and direct preference optimization (DPO) [143] ensure that agent behavior conforms to human values and domain-specific norms.

All evolved artifacts are versioned through the MLflow Model Registry [126] to ensure traceability and enable safe rollback when necessary. Systematic evaluation employs controlled experiments such as A/B testing [167] to compare agent variants across dimensions including task success rate, safety compliance, and user satisfaction. The results of these evaluations inform data-driven decisions regarding promotion to production.

Together, these four phases constitute a resilient and adaptive lifecycle model that empowers agentic services to operate sustainably, evolve intelligently, and maintain trustworthiness in complex, open, and dynamic environments.

IV. AGENTIC SERVICES COMPUTING: THE RESEARCH FRAMEWORK

A. Redefining Services Computing in the Age of Agency

The era of services computing is undergoing a tectonic shift. Where once we orchestrated stateless functions, we now coordinate intentional agents. Where once we modeled linear workflows, we now design dynamic societies of collaborating entities. And where once we optimized for correctness and latency, we must now guarantee *trustworthiness*, *alignment*, and *accountability* in open, evolving environments. ASC is not an incremental extension of classical service models, but a foundational reimagining. ASC positions intelligent agents not as wrappers around APIs, but as *first-class, evolvable services* with lifecycles, responsibilities, and social contracts. It replaces brittle, one-shot invocations with adaptive, context-aware agency that persists, learns, and collaborates across organizational and technical boundaries.

Definition 1 (Agentic Services Computing): Agentic Services Computing is a lifecycle- and capability-driven paradigm for

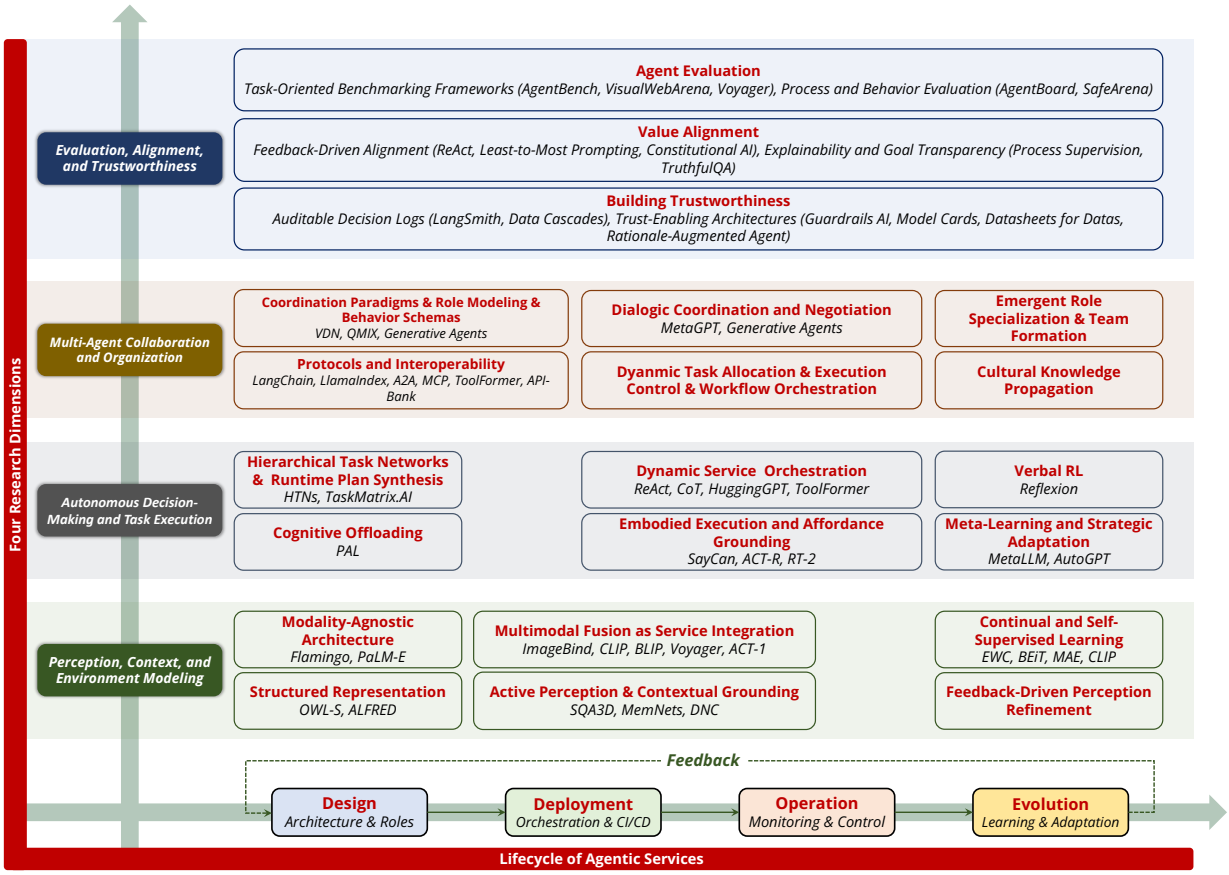

Figure 4: Research framework of Agentic Services Computing, mapping core research dimensions across the service lifecycle. Each cell highlights key technologies and research topics.

engineering intelligent agents as first-class services. It enables agentic services to perceive multimodal environments, make autonomous decisions, collaborate within organizational contexts, and evolve through continuous evaluation, while ensuring alignment, transparency, and trustworthiness throughout their operational lifespan.

B. The ASC Framework: A Matrix of Agency and Evolution

As illustrated in Figure 4, the ASC framework is organized along two orthogonal axes that together define the space of next-generation service engineering:

- *Horizontal axis* lies the agentic service lifecycle, including DESIGN, DEPLOYMENT, OPERATION, and EVOLUTION. This axis captures the full temporal arc of an agentic service, from specification to retirement.
- *Vertical axis* lies four pillars of agentic capability, including: (i) *Perception, Context, and Environment Modeling*, (ii) *Autonomous Decision-Making and Task Execution*, (iii) *Multi-Agent and Organizational Collaboration*, and (iv) *Evaluation, Alignment, and Trustworthiness*.

Unlike classical service stacks, ASC treats these dimensions as *co-evolving*: each lifecycle phase deepens engagement across all four pillars. In DESIGN, agents are grounded in semantic world models (e.g., OWL-S [118], ALFRED [157]) and hierarchical planners (HTNs [65]). In DEPLOYMENT, they acquire tool access via interoperability standards like MCP [16] and ToolFormer [152], and instantiate collaboration topologies through frameworks like LangGraph [14]. During OPERATION, they perform real-time contextual grounding [139], [110], dynamic orchestration [195], [155], and auditable reasoning [15], [47]. Finally, in EVOLUTION, they refine themselves through self-supervised learning [89], [30], meta-adaptation [129], and human-aligned feedback [195], [203].

To appreciate ASC's transformative potential, consider three fundamental contrasts:

- *From functions to agents.* Classical services are passive: invoked, executed, forgotten. ASC services are active: they observe, decide, initiate, and persist. They carry memory, intent, and responsibility.
- *From workflows to societies.* Traditional orchestration assumes centralized control and deterministic paths. ASC embraces decentralized, emergent coordination, where teams of agents self-organize [162], [145], negotiate roles, and resolve conflicts under shared norms.
- *From correctness to trustworthiness.* Correctness asks: "Did the service return the right output?" Trustworthiness asks: "Can we understand why? Was it fair? Is it safe to rely on it tomorrow?" ASC embeds interpretability, alignment, and auditability into the service fabric.

In each case, ASC transforms agents from fragile, opaque tools into accountable, evolvable participants in socio-technical systems.

C. The Milestones

Realizing ASC demands more than algorithms. It requires new infrastructure, standards, and governance primitives. We propose a three-phase roadmap:

- *Short-term (1–2 years): Lifecycle-aware observability.* Build mandatory agent registries (for prompts, policies, memory schemas), standardized audit APIs that expose reasoning-action traces, and runtime monitors. These form the bedrock of reproducibility, debugging, and compliance.
- *Mid-term (2–4 years): Agent-governed contracts.* Formalize ASLOs as machine-enforceable service level objectives covering alignment, safety, interpretability, and sustainability (e.g., energy-per-decision). Integrate reputation systems and safe retirement protocols to enable dynamic, open-market composition.
- *Long-term (5+ years): Interoperable agent societies.* Achieve ecosystems where heterogeneous agents that built by different vendors, governed by different constitutions can discover, negotiate, and co-evolve under shared meta-governance. This requires standardized ontologies for intent exchange, cross-registry identity, and constitutional interoperability.

ASC is more than a research agenda; it is a call to rebuild services computing for an age of agency. The following sections dissect the four pillars of agentic capability, analyzing their technical foundations and the unique challenges they pose at each stage of the agent lifecycle. Together, they form the blueprint for a new generation of intelligent, responsible, and resilient digital services. The following sections elaborate on each of the four research dimensions in the ASC framework, analyzing their technical underpinnings and the distinct challenges they pose at each stage of the agentic service lifecycle.

V. Perception, Context, and Environment Modeling

Effective operation in real-world environments requires agentic services to perceive and model their surroundings with semantic coherence. This environment modeling underpins all decision-making, planning, and interaction. Unlike traditional agents that operate in symbolic domains, modern agentic services operate in open-ended, dynamic, and multimodal settings. These settings include physical spaces equipped with sensors as well as digital ecosystems governed by social dynamics. Such environments demand perception mechanisms that are both robust and adaptive. Environment modeling is not a one-time design artifact. Rather, it constitutes a cross-cutting concern that must be integrated throughout the service lifecycle. During the Design phase, perception architectures are co-developed with functional and non-functional requirements to ensure semantic alignment. In the Deployment phase, context-aware models are instantiated under real-world constraints, including latency, privacy, and resource availability. Throughout the Evolution phase, these models are continuously monitored and updated to address concept drift, accommodate new modalities, and respond to shifts in user behavior. Figure 5 illustrates the key dimensions of this modeling process.

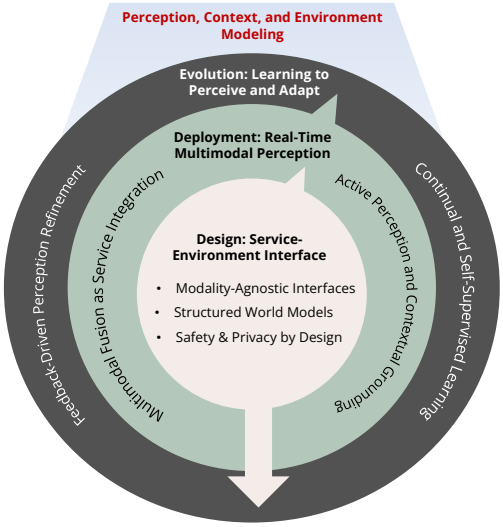

Figure 5: Key dimensions of perception, context, and environment modeling in agentic services.

A. Design: Modeling the Service-Environment Interface

The Design phase establishes the agentic service's epistemic boundaries by defining sensory modalities, such as text, vision, audio, and sensor streams, along with observation spaces and assumptions about environmental dynamics. These dynamics include partial observability, noise, and temporal coherence. This foundational stage shapes the agentic service's capacity to interpret stimuli and ground decisions in real-world contexts.

1) Decoupling Perception and Reasoning in Modality-Agnostic Architectures: A key trend in agentic service design is the adoption of *modality-agnostic* architectures that decouple perception from reasoning. This separation supports modularity and reusability. Flamingo [25] employs a perceiver resampler to condition a frozen LLM on interleaved images and text. This approach enables few-shot multimodal reasoning without retraining the core language model. Similarly, PaLM-E [62] fuses visual and robotic inputs into an LLM to support embodied planning. This architectural pattern enables *perception pluggability*, which allows developers to replace sensor stacks without modifying the reasoning logic.

2) Structured World Representations for Grounded Reasoning: The design phase also involves constructing semantically rich world models to support grounded reasoning. Knowledge graphs encode domain-specific facts and relational constraints [92]. When these graphs are aligned with formal frameworks such as OWL-S [118], they facilitate machine-understandable service composition. Spatial memory maps and neural scene representations are used to support navigation and manipulation tasks [82]. The ALFRED benchmark [157] formalizes complex household tasks in simulated environments with visual and linguistic inputs. It serves both as an evaluation platform and as a design pattern for goal-oriented, contextually grounded agents.

3) Designing for Safety, Privacy, and Uncertainty: Non-functional requirements must be embedded during the design phase. Agents may be restricted from processing biometric data or required to anonymize inputs using techniques such

Table II: Key multimodal perception models in agentic services. For modalities, **t**=Text, **i**=Image, **s**=Sensors, **c**=Code, **u**=UI, **ac**=Action.

Model	Modalities	Core Mechanism	Service Interface
Flamingo [25]	**t**, **i**	Perceiver Resampler + LLM	API-style input
PaLM-E [62]	**t**, **i**, **s**	Embodied LLM fusion	Task-level commands
CLIP [139]	**i**, **t**	Contrastive pretraining	Zero-shot classifier
BLIP [111]	**i**, **t**	Generative + contrastive fusion	Captioning/QA API
ImageBind [80]	all included	Zero-shot alignment	Embedding service
VOYAGER [177]	**i**, **c**, **t**	Iterative prompting + skill library	Minecraft command API
MobileVLM [189], [49]	**u**, **t**	Mobile UI grounding	App automation API
ACT-1 [24]	**u**, **t**, **ac**	Imitation learning + API calls	App interaction engine

as differential privacy [22]. These practices align with regulatory frameworks like the GDPR [173] and the FAIR principles [184]. Policy annotations and access control mechanisms enforce *compliance-by-design*, which ensures that ethical and legal constraints are integral to the system architecture. Additionally, uncertainty modeling is implemented through Bayesian networks or confidence scoring, and it supports robust decision-making under conditions of partial observability.

B. Deployment: Real-Time Multimodal Perception and Service Activation

In the DEPLOYMENT phase, agentic services process multimodal inputs under conditions of uncertainty, partial observability, and strict latency constraints. Robustness, efficiency, and contextual grounding are critical to meeting SLAs, ensuring a positive user experience, and maintaining system reliability. Table II summarizes key multimodal perception models and their deployment characteristics.

1) Multimodal Fusion as Service Integration: Integrating heterogeneous data relies on established fusion strategies. Early fusion combines raw inputs before feature extraction. This approach captures low-level correlations but requires precise synchronization, which is often impractical in distributed systems. Late fusion processes each modality independently and then combines decisions at the output level. It enhances modularity and fault isolation, making it suitable for microservice-based architectures. Cross-modal attention enables dynamic interaction between modalities. Models such as CLIP [139] and BLIP [111] learn joint image-text embeddings, supporting zero-shot retrieval and semantic grounding without retraining. Modality translation aligns disparate inputs into a shared latent space. ImageBind [80] embeds six modalities, which are image, text, audio, depth, thermal, and IMU, into a unified representation, enabling zero-shot transfer when certain sensors are unavailable. Frameworks such as VOYAGER [177], ACT-1 [24], and MobileVLM [49], [189] integrate these models into production systems. These frameworks are optimized for edge deployment through techniques such as model quantization, embedding caching, and adaptive sampling.

2) Active Perception and Contextual Grounding: Passive perception is insufficient for complex tasks. Agents employ *active perception* to gather informative observations. This includes asking clarifying questions, adjusting camera angles, zooming into image regions, or requesting additional context from users [55]. Active perception transforms agents into proactive information seekers, enhancing robustness in ambiguous environments. Contextual grounding enables agents to interpret perceptions within the broader task, social, and environmental context. The SQA3D benchmark [117] requires agents to answer questions using spatial and temporal context. In human-agent interaction, pragmatic understanding allows agents to infer intent from tone, gesture, and conversational history [26]. Memory-augmented architectures such as MemNets [164] and Differentiable Neural Computers (DNCs) [83] store and retrieve past interactions. This capability supports coherence across sessions, enables counterfactual reasoning, and facilitates experience-based learning. These mechanisms form the foundation for long-term adaptation and continuous evolution.

C. Evolution: Learning to Perceive and Adapt

In the EVOLUTION phase, agents refine their perception through experience, feedback, and environmental interaction. This process transforms perception from a static design artifact into an adaptive, living component of the service lifecycle, thereby enabling long-term resilience in dynamic real-world settings.

1) Continual and Self-Supervised Learning: To sustain performance under distribution shifts, such as the appearance of novel objects, changes in lighting conditions, or evolving user demographics, agentic services employ continual learning strategies that balance plasticity with stability. Elastic Weight Consolidation (EWC) [104] protects critical weights associated with prior tasks, while experience replay [149] revisits representative samples from past environments. Both approaches mitigate catastrophic forgetting and enable incremental model updates without requiring full retraining. Complementing continual learning, self-supervised learning reduces reliance on costly labeled data. Models such as BEiT [30] and MAE [89] leverage masked image modeling to pre-train visual encoders on unlabeled data, achieving strong downstream performance with minimal fine-tuning. In multimodal contexts, contrastive frameworks like CLIP [139] enable large-scale pre-training on web-

scraped image-text pairs, supporting zero-shot transfer and rapid adaptation to new domains. Together, these approaches form the foundation for lifelong perceptual learning, allowing agents to expand their sensory understanding autonomously.

2) Feedback-Driven Perception Refinement: Real-world deployment generates rich signals that support iterative refinement. Agents exploit both explicit and implicit feedback to correct errors, close performance gaps, and drive higher-level adaptation:

- *Human-in-the-loop correction*: Users provide direct corrections (e.g., "That's not the file I meant"), which are logged and used for targeted retraining [190]. This closed-loop interaction enhances accuracy, personalization, and user trust.
- *Uncertainty-aware escalation*: Anomaly detection identifies low-confidence or out-of-distribution inputs [90], triggering human review or fallback behaviors. This ensures robustness in safety-critical scenarios while capturing edge cases for future learning.
- *A/B testing and telemetry*: Controlled experiments compare perception variants, such as object detectors or fusion strategies [167]. Task failure logs and engagement metrics reveal persistent blind spots, including challenges in occlusion handling or modality misalignment, which in turn prompt architectural improvements such as enhanced sensor integration or redesign of fusion mechanisms.

This feedback-driven adaptation not only improves perceptual accuracy but can also initiate broader evolutionary changes. Such changes may include redesigning the service-environment interface (discussed in Section V-A) or reconfiguring active perception policies during deployment (discussed in Section V-B).

D. Summary and Challenges

Perception and environment modeling constitute the cognitive foundation of agentic services across the entire lifecycle: from the DESIGN of sensory interfaces and world representations, through the DEPLOYMENT of real-time multimodal processing, to EVOLUTION via experience and feedback. Despite advances in fusion architectures, active perception, and self-supervised learning, significant gaps remain between controlled development environments and the complexity of open-world settings. We identify three cross-cutting challenges that define the frontier of future research:

1) *Generalizable Contextual Representation.* Perception models trained under narrow assumptions often fail when deployed in novel or culturally diverse environments. The challenge lies in moving beyond surface-level statistical patterns toward semantically grounded, invariant representations, such as object affordances, spatial hierarchies, physical dynamics, and social norms. Achieving robust generalization requires integrating structured priors (e.g., ontologies, physics engines, commonsense knowledge bases) into the architecture during design, thereby enabling zero-shot transfer and contextual adaptability.
2) *Robustness to Environmental Uncertainty.* Agents face sensor noise, occlusions, and partial observability, with individual modalities degrading unpredictably. For example, cameras in low-light conditions or microphones in noisy settings. Although multimodal fusion improves resilience, many systems remain brittle. The key challenge is to design adaptive perception pipelines that dynamically assess uncertainty, reweight modalities according to reliability, and invoke graceful fallbacks. Addressing this challenge requires tight integration of uncertainty quantification, active perception, and fault-tolerant mechanisms.
3) *Continual Environmental Understanding.* Environments evolve: new objects emerge, interaction patterns shift, and task goals change over time. Static models suffer from semantic drift and performance decay. The challenge is to support lifelong learning, where agents continuously update their perceptual and contextual knowledge through feedback, exploration, and anomaly detection [177], while balancing plasticity with stability. This demands learning algorithms that resist catastrophic forgetting and architectures that support modular, interpretable updates.

VI. AUTONOMOUS DECISION-MAKING AND TASK EXECUTION

Autonomous decision-making and task execution constitute the behavioral core of agentic services, transforming high-level intent into concrete actions within complex environments. While perception establishes the input boundary, agents manifest their agency through reasoning, planning, and adaptive execution. Modern agentic systems no longer rely on rigid, prescripted workflows; instead, they dynamically interleave cognition and action, recover from failures, and reconfigure strategies in response to environmental feedback. During the OPERATION phase, agents function as self-governing orchestrators, composing modular tools, grounding plans in physical or digital affordances, and enforcing runtime resilience. In the EVOLUTION phase, execution strategies are refined through experience, enabling adaptation to novel tasks and changing conditions. The (RE)DESIGN phase incorporates insights from operational failures and environmental shifts to restructure reasoning pipelines, tool interfaces, or recovery protocols. This transformation closely parallels the design principles of cloud-native and microservices architectures [127], [148]. Figure 6 illustrates the fundamental paradigms and mechanisms that underpin this execution layer.

A. Operation: Dynamic Service Orchestration and Real-Time Execution Control

In services computing, OPERATION ensures reliable delivery through state management, failure recovery, and SLA compliance. For autonomous agents, this operational paradigm manifests as multistep task execution under uncertainty, where reasoning, tool use, and environmental interaction are tightly coupled within a continuous control loop. Unlike static workflows, agents function as self-governing orchestrators that dynamically coordinate actions, adapt to feedback, and maintain contextual awareness. Four interdependent pillars underpin robust execution: (i) reasoning-action interleaving for adaptive planning, (ii) modular tool composition as dynamic service integration, (iii) embodied grounding to ensure physical feasibility, and (iv) fault-tolerant mechanisms for resilient operation. Together,

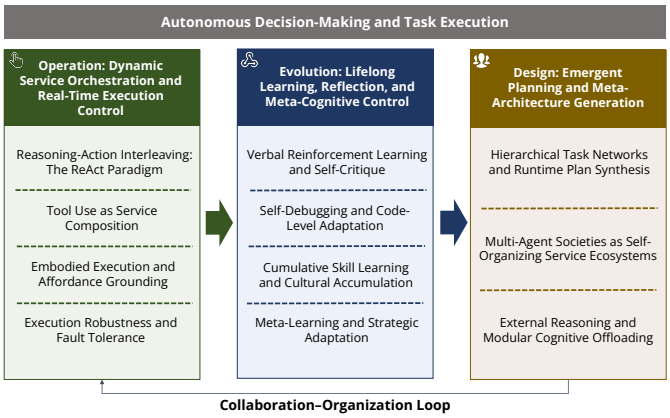

Figure 6: Core paradigms and mechanisms for the autonomous decision-making and task execution.

these components constitute a runtime control architecture that mirrors the principles of resilient distributed systems, enabling agents to operate reliably in unpredictable environments.

REACT [195] interleaves natural language reasoning with external actions, thereby enabling adaptive planning grounded in real-world observations. In contrast to static Chain-of-Thought (CoT) prompting [183], which produces a monolithic reasoning trace prior to any action, REACT revises its strategy iteratively based on environmental feedback, emulating the principles of model-based reinforcement learning [163]. Formally, let $\pi_\theta(s_t) \to (a_t, r_t)$ denote a policy that, given state s_t, generates an action a_t and a reasoning trace r_t. The environment returns observation o_{t+1}, updating both state and reasoning via:

- State transition: $s_{t+1} = f(s_t, a_t)$,
- Reasoning update: $r_{t+1} = g(s_{t+1}, \mathcal{K})$, where $\mathcal{K}$ is background knowledge.

This closed loop implements real-time control, supporting recovery, exploration, and incremental refinement, which are key to agile operation in unpredictable domains.

1) Tool Use as Service Composition: Agents treat external tools as modular, composable services. HuggingGPT [155] exemplifies this approach by employing a LLM as a meta-controller that routes tasks across specialized models such as BLIP [111] for vision, Whisper [140] for speech, and Stable Diffusion [150] for generation, thereby realizing a Function-as-a-Service (FaaS) architecture [29]. The LLM assesses task requirements, selects appropriate services, sequences their invocation, and synthesizes the results, effectively functioning as an intelligent orchestration engine. This paradigm differs fundamentally from predefined workflow systems like Apache Airflow [1] or Kubernetes [5], where orchestration logic is statically encoded. In contrast, agentic orchestration is generated on demand through semantic understanding of the task context. ToolFormer [152] further advances this capability by learning when to invoke tools, such as calculators or question-answering APIs, through *self-supervised prompting*, thereby acquiring a latent policy for service invocation. This shift from static pipelines to dynamic, just-in-time composition enables agents to respond flexibly to novel and unforeseen tasks.

2) Embodied Execution and Affordance Grounding: Successful execution in physical or simulated environments requires grounding high-level commands in perceptual and physical constraints, a challenge known as the symbol grounding problem [87]. SayCan [62] addresses this by decoupling semantic planning from feasibility assessment: a language model proposes semantically coherent action sequences (e.g., "bring me a glass of water"), while a separate value model estimates the likelihood of successful execution for each candidate action given the agent's current capabilities and environmental context. Only actions with high feasibility scores are executed, ensuring compatibility with real-world affordances. This architectural separation mirrors cognitive frameworks such as ACT-R [147], which distinguish between declarative knowledge (what to do) and procedural knowledge (how to do it). Building on this, RT-2 [205] integrates language and vision inputs into an end-to-end model that maps directly to robot motor controls, thereby bridging symbolic reasoning and physical action. These approaches collectively narrow the gap between abstract intent and embodied execution.

3) Execution Robustness and Fault Tolerance: Operation in noisy and dynamic environments necessitates resilient execution strategies. Agents employ adaptive recovery mechanisms that go beyond simple retries: they rephrase queries upon search failure [195], switch to alternative tools when primary services are unavailable, and parse structured error messages, such as HTTP 404 responses or rate-limit headers, to infer corrective actions. These behaviors parallel established resilience patterns in cloud-native systems, including circuit breakers, timeout policies, and graceful degradation. Furthermore, agents leverage episodic and semantic memory to avoid repeating past failures, thereby implementing a form of *learned* fault tolerance. This transition from passive error handling to active recovery positions agents as self-healing endpoints within service ecosystems.

B. Evolution: Lifelong Learning, Reflection, and Meta-Cognitive Control

Unlike traditional software systems that rely on periodic, externally driven updates, autonomous agents enable continuous, self-directed evolution through internal feedback loops. This capacity supports long-term adaptation in open-ended environments by integrating mechanisms such as verbal self-reflection [156], code-level self-debugging [72], cumulative skill acquisition [177], and meta-strategic adjustment [161]. Through these processes, agents transcend the role of static service endpoints and emerge as self-evolving entities that learn from experience, refine their behavior, and autonomously expand their capabilities over time.

1) Verbal Reinforcement Learning and Self-Critique: Verbal reinforcement learning [156] leverages natural language critiques generated after task failure, such as "I assumed the flight was domestic", to construct interpretable, context-sensitive reward signals. These critiques are stored in semantic memory and retrieved during subsequent task attempts, forming a dynamic reward model that evolves with experience. This approach offers three key advantages: first, it enhances interpretability by producing human-readable feedback; second, it enables precise error diagnosis by linking failures to specific reasoning steps or

assumptions; third, it facilitates cross-task transfer, as lessons learned in one domain, e.g., handling date formats in travel planning, generalize to related tasks like calendar management or document parsing. By transforming failures into structured learning opportunities, verbal RL fosters a form of operational self-awareness that mirrors reflective practice in human expertise.

2) Self-Debugging and Code-Level Adaptation: Agents are capable of detecting and repairing code-level errors autonomously. Systems such as REACT [195] and APRMCTS [68] implement closed-loop debugging cycles in which generated code is executed, runtime exceptions are parsed, and fixes are proposed through in-context learning, iterating until successful execution is achieved. Critically, large language models infer the intended semantics behind the code, enabling them to correct deep logical errors that extend beyond syntactic mistakes. This represents a significant departure from classical automated program repair (APR) techniques [119], which rely on syntactic mutation and search heuristics without modeling high-level intent. From a systems perspective, this capability allows agents to function as self-healing components [102], a feature that is especially valuable in edge or distributed computing environments [200] where external intervention is impractical and resilience must be internalized.

3) Cumulative Skill Learning and Cultural Accumulation: True intelligence entails not only adaptation but the structured accumulation and reuse of knowledge across tasks and time. VOYAGER [177] demonstrates this principle in the open-ended environment of Minecraft, where sustained exploration and procedural mastery are required. The agent maintains a versioned skill library composed of executable functions (e.g., `craft_iron_pickaxe()`), which are discovered through trial and error, refined via repeated use, and composed into increasingly complex behaviors. A curriculum agent proposes new skills based on exploration progress, while an instructor agent generates natural language descriptions that specify usage conditions and compositional rules. This process mirrors cumulative cultural evolution [170], wherein knowledge compounds across iterations. By institutionalizing experience into reusable, evolvable components, agents shift from reactive execution toward proactive capability building. This model aligns with SOA principles of modularity and reuse, now extended into the cognitive domain where skills function as versioned, composable services.

4) Meta-Learning and Strategic Adaptation: To operate effectively across diverse and novel domains, agents must not only learn new tasks but also improve their learning process itself. This is a capability known as meta-learning. Frameworks such as MetaLLM [129] and AutoGPT [10] implement this through meta-prompts that encode higher-order cognitive strategies, such as "Break the problem into subgoals" or "If stuck, search for examples." These prompts serve as internal heuristics that dynamically adjust the agent's reasoning trajectory in response to task complexity, ambiguity, or feedback. This embodies the principle of *learning to learn* [94], where adaptation occurs at the algorithmic level rather than merely at the output level. Functionally, these mechanisms resemble self-configuring middleware: the agent autonomously tunes its cognitive policy based on task demands, resource constraints, or interaction patterns. For instance, it may activate hierarchical decomposition for complex planning tasks or invoke clarification protocols when faced with ambiguous instructions. Such meta-cognitive control is essential for scalable, general-purpose agents that must operate across heterogeneous environments without manual reconfiguration.

C. Design: Emergent Planning and Meta-Architecture Generation

Traditional service design is static and pre-deployment. Agentic services perform *runtime design*: dynamically decomposing tasks, synthesizing plans, and reconfiguring cognition in response to environmental demands. This enables flexible, goal-driven behavior in open domains. Agents act as self-designing entities, composing and coordinating functional components at runtime. Three key mechanisms underpin this capability: hierarchical task decomposition, multi-agent self-organization, and modular cognitive offloading.

1) Hierarchical Task Networks and Runtime Plan Synthesis: Agents implicitly realize Hierarchical Task Networks (HTNs) [65] through language-driven goal decomposition. Systems like REACT [195] and TaskMatrix.AI [113] translate high-level goals, e.g., "Plan a birthday party", into executable subtasks such as venue search or email drafting, dynamically invoking external tools. Plans adapt in real time: if a venue is unavailable, new subtasks are generated on the fly. The result is a dynamic workflow graph that enables autonomous, adaptive service composition, which is similar to event-driven architectures enhanced with semantic reasoning.

2) Multi-Agent Societies as Self-Organizing Service Ecosystems: Design can also emerge from structured interaction among specialized agents. MetaGPT [93] simulates a software startup with roles (CEO, PM, Engineer, QA), each communicating via standardized messages (e.g., specs, code reviews). The CEO sets goals; the PM refines them; the Engineer implements; QA validates. This mirrors service-oriented organizations [134], where loosely coupled services interact via contracts. The system architecture emerges from roles and norms, not top-down specification, enabling scalability, fault isolation, and organizational learning. It foreshadows agent-based enterprises capable of autonomously managing full software lifecycles.

3) External Reasoning and Modular Cognitive Offloading: Separating reasoning from execution enhances reliability and auditability. The Program-Aided Language (PAL) framework [74] offloads symbolic computation (e.g., arithmetic, data transforms) to an external interpreter like Python, while the LLM handles planning and language. This hybrid neuro-symbolic pipeline ensures precision without sacrificing flexibility. Architecturally, it embodies "cognitive microservices": modular, composable intelligence units that encapsulate distinct capabilities (planning, computation, tool use) and can be independently scaled, versioned, and orchestrated at runtime.

D. Summary and Challenges

Agentic services leverage dynamic reasoning, adaptive composition, and emergent design to autonomously orchestrate and

evolve service workflows. Agents no longer merely consume services; they actively design, coordinate, and refine them at runtime. However, as these systems transition from research prototypes to production-grade infrastructures, their autonomy introduces critical challenges in predictability, reliability, and maintainability. We identify three core challenges aligned with key phases of the service lifecycle:

1) *Emergent Planning with Predictable Outcomes.* Dynamic planning via hierarchical decomposition, role negotiation, or multi-agent consensus enables flexibility but often yields inconsistent, redundant, or unverifiable workflows. The challenge is achieving *guided emergence*: supporting adaptive planning while respecting safety, budget, and intent constraints. This requires (i) *plan traceability*, linking goals to executable steps, (ii) *versioned workflows* for rollback and A/B testing, and (iii) *interoperability* with standards like OpenAPI or BPMN for enterprise integration.
2) *Robust Execution in Dynamic Service Environments.* Agents face volatile conditions in deployment, shifting APIs, network failures, or semantic drift in tool affordances. Yet they lack systematic fault isolation, transactional semantics, or graceful degradation. The challenge is *principled fault tolerance*: modeling actions as stateful services with clear pre/postconditions and side-effect contracts, and integrating microservices patterns such as circuit breaking, idempotent retries, and compensating transactions to ensure reliability at scale.
3) *Cumulative Learning without Knowledge Entropy.* As agents accumulate skills, reflections, and code over time, they risk *knowledge entropy*: redundancy, drift, conflicting behaviors, and inefficient retrieval. Sustainable learning demands active curation through (i) *knowledge distillation* [91] to compress memories into compact policies, (ii) *modular abstraction* to extract reusable functions, and (iii) *consistency validation* to resolve behavioral conflicts. Without these, cognitive overhead can undermine long-term autonomy.

VII. Multi-Agent Collaboration and Organization

As service systems grow in complexity, single-agent architectures struggle with open-ended, dynamic tasks. Effective solutions increasingly rely on multi-agent collaboration, where agents coordinate through structured interaction, knowledge sharing, and adaptive behavior. This section analyzes collaboration across three lifecycle phases: Design, Operation, and Evolution. We argue that successful collaboration requires more than task decomposition; It demands role specialization, shared norms, and intentional architecture. While deployment and retirement support continuity, core collaborative dynamics emerge during design, execution, and long-term adaptation. Figure 7 outlines key paradigms and mechanisms.

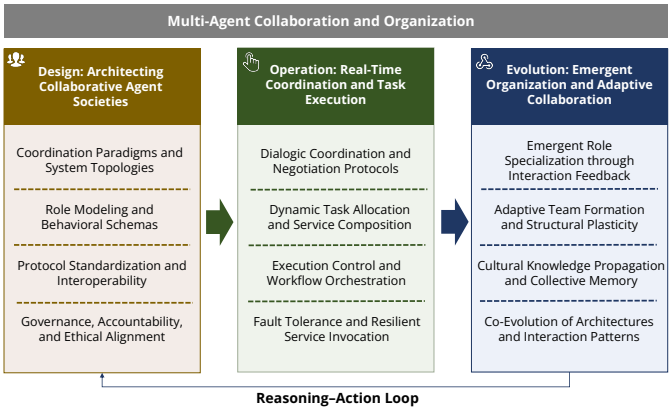

Figure 7: Core paradigms and mechanisms for multi-agent collaboration.

A. Design: Architecting Agent Interaction

The Design phase establishes the structural foundation of multi-agent systems, defining topologies, roles, protocols, and governance. As these systems evolve into open ecosystems, intentional architecture becomes essential for scalability, robustness, and trust.

1) Coordination Topologies: Three primary models shape system behavior:

- *Centralized orchestration* uses a single controller (e.g., AutoGen [188]) to assign tasks and resolve conflicts. It ensures consistency but risks bottlenecks and single points of failure [72].
- *Decentralized negotiation* enables peer-to-peer coordination via dialogue or auctions [79], improving resilience and scalability at the cost of higher communication overhead and potential inefficiency.
- *Hybrid hierarchies* combine both models dynamically. For example, Magentic-One [72] employs planner-executor hierarchies for structured workflows but allows local negotiation when obstacles arise, balancing control with adaptability.

Frameworks like AgentSociety [138] support configurable topologies, enabling performance trade-offs in domains such as healthcare and disaster response.

2) Roles and Behavioral Schemas: Roles define functional responsibilities, such as `Planner` or `Critic`, as well as interaction patterns. Static roles provide predictability, whereas dynamic roles adapt to contextual changes or performance feedback, thereby enhancing flexibility. Multi-Agent Reinforcement Learning (MARL) methods, including VDN [162] and QMIX [145], facilitate emergent role specialization by optimizing global rewards under decentralized execution policies. Beyond functional assignment, behavioral schemas, e.g., turn-taking, politeness norms, or conflict resolution strategies, reduce coordination friction and promote social coherence. In *Generative Agents* [136], memory, reflection mechanisms, and scripted behaviors enable agents to produce socially plausible interactions. This demonstrates that normative behavior can emerge when grounded in structured cognitive models and sustained environmental feedback.

3) Protocol Standardization and Interoperability: As multi-agent ecosystems transition from closed systems to open, heterogeneous networks, interoperability becomes a foundational design challenge. Without shared communication standards, data

Table III: Comparison of Agent Interoperability Protocols. Columns: A2A (Agent-to-Agent), MCP (Model-Context-Protocol), FIPA (Agent Communication Language), OpenAI FC (Function Calling). Rows: Model = interaction paradigm; Msg. Format = message syntax; Stateful = persistent context support; Auth/NID = identity/authentication; Tool Support = interoperability with external tools; Discovery = capability/service discovery; Semantics = level of shared meaning; Use Case = typical deployment scenario. ✓ = supported, ✗ = not or limited. *: authentication via API key.

Feature	A2A [11]	MCP [16]	FIPA [71]	OpenAI FC [17]
Model	A2A dialogue	Resource-oriented	Speech-act ACL	Function call
Msg. Format	JSON (streaming)	JSON (typed)	XML/SL	JSON Schema
Stateful	✓	✓	✗	✗
Auth/NID	✓	✗	✗	✗*
Tool Support	Indirect	✓	Limited	✓
Discovery	`/spec`	Context-aware	Yellow Pages	Manual
Semantics	High	High	High	Low
Use Case	Secure collaboration	Data workflows	Academic MAS	LLM tool use

models, and behavioral expectations, agents cannot collaborate effectively or compose into higher-order services. Standardization efforts address four key dimensions: (i) semantic communication, (ii) message schemas, (iii) interaction protocols, and (iv) tool interoperability. Table III compares representative protocols across these dimensions.

- *Semantic communication standards.* The FIPA initiative [71] pioneered agent interoperability through its Agent Communication Language (ACL), which is grounded in speech act theory. FIPA-ACL standardized performatives such as *request*, *inform*, and *propose*, with content encoded in formal languages like SL. Although influential in academic multi-agent systems research, FIPA's complexity and misalignment with modern web practices have limited its real-world adoption.
- *Message schema and payload standards.* Contemporary frameworks, including LangChain [14], LlamaIndex [6], and Semantic Kernel [9], adopt lightweight, JSON-based message formats inspired by REST APIs. These schemas typically include fields such as *role*, *content*, *tool_calls*, and *tool_results*, which enable basic cross-platform compatibility. However, they lack formal semantics and persistent context modeling, which reduces their reliability in long-running or multi-turn collaborations.
- *Interaction protocol layers.* Recent advances introduce structured protocol layers for agent-to-agent (A2A) interaction. The A2A protocol [11] defines a REST-like API for secure, stateful communication. It supports capability discovery via `GET /spec`, conversation initiation with `POST /messages`, streaming of responses and tool calls, and standardized error handling. A2A emphasizes agent identity, authentication, and intent clarity, which enables trust and provenance tracking. Its support for asynchronous workflows and callbacks makes it suitable for complex, long-term tasks.
  In parallel, the Model-Context-Protocol (MCP) framework [16] proposes a layered architecture, as illustrated in Figure 8. The *Model* layer captures shared domain semantics, such as those of a CRM system or simulation environment. The *Context* layer maintains task-specific state, including interaction history and collaborative goals. The *Protocol* layer defines standardized operations, including `read()`, `write()`, `request()`, and `subscribe()`. MCP extends REST principles with typed, persistent context, thereby facilitating dynamic composition across heterogeneous domains, particularly in enterprise data integration.
  While A2A prioritizes secure dialogue and agent identity, MCP focuses on resource-oriented, stateful interactions. Their differing emphases suggest future standards could integrate both: using A2A for discovery and negotiation, and MCP for deep, context-aware resource access.
- *Tool and Function Interoperability.* Standardized tool interfaces are critical for service composition. The OpenAI Function Calling schema [17], which is based on JSON Schema, has become a de facto standard for structuring tool requests from LLMs. It is widely adopted in Vertex AI [81], Copilot Studio [122], and open-source platforms such as Haystack [13] and LangGraph [14], enabling consistent tool invocation across ecosystems. Research efforts such as Toolformer [152] and API-Bank [112] explore agents' ability to autonomously learn and use APIs, underscoring the need for machine-readable and semantically rich spec-

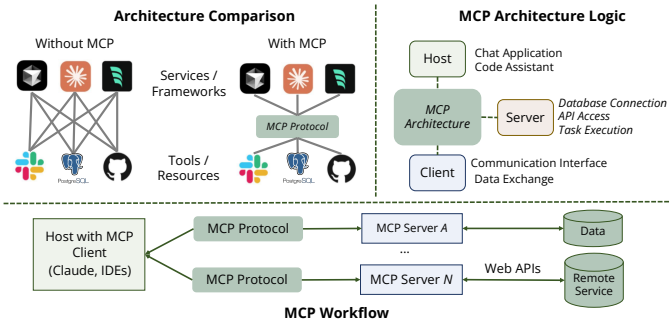

Figure 8: The workflow of the MCP framework.

ifications. The broad adoption of OpenAPI for API documentation signals significant progress toward universal tool discoverability and self-service integration in agent ecosystems.

4) Governance, Accountability, and Ethical Alignment: As agentic systems grow in autonomy and complexity, robust governance must be operationalized through concrete, lifecycle-aware mechanisms. In decentralized environments, where outcomes emerge from distributed interactions, traditional oversight models fail to answer foundational questions: Who bears responsibility when a team of agents causes harm? How are norm violations detected, attributed, and remediated?

To address these challenges, we call for a *lifecycle governance toolkit* that embeds accountability into every phase of an agentic service's existence. First, *agent registries* serve as mandatory, version-controlled repositories for an agent's core artifacts (e.g., prompts, memory schemas, policy constraints, and tool bindings, etc.), functioning as a "GitHub for agents". Such registries enable reproducibility, provenance tracking, and rollback in case of failure or drift. Second, *retirement protocols* define safe shutdown procedures that deactivate agents, purge sensitive state, and revoke access rights, thereby preventing the emergence of unmonitored "zombie agents" that persist beyond their operational mandate. Third, *audit APIs* require every agentic service to expose structured, interpretable logs that capture not only executed actions but also the reasoning traces behind them, enabling external verification, debugging, and compliance checks. These mechanisms extend familiar concepts from services computing into the agentic era. SLAs are generalized into *Agent Service Level Objectives* (ASLOs), which specify measurable commitments not only on latency or availability but also on alignment fidelity, interpretability thresholds, and safe action boundaries. For instance, an ASLO might constrain an agent to reject requests that violate a constitutional principle or to escalate decisions involving personal data to human review.

Emerging architectural patterns support this vision. Decentralized Autonomous Organizations (DAOs) [166] use smart contracts and on-chain voting to enable transparent, rule-based coordination among agents. While promising, DAOs require secure identity anchoring and defenses against manipulation, such as Sybil attacks. Complementary reputation systems like TrustChain [20] assign dynamic trust scores based on interaction history, informing behavior-based access control and role delegation, making agent conduct observable and its consequences tangible. For high-stakes scenarios, human-in-the-loop governance [165] ensures ethical alignment by escalating policy ambiguities or moral dilemmas to human reviewers, preserving judgment where automated reasoning is insufficient. Proactive safety is further reinforced through a *safety-by-design* approach. Runtime monitors, behavioral sandboxes, and action constraints limit the blast radius of unintended or malicious behavior. The Constitutional AI framework [28] exemplifies this by equipping agents with a normative "constitution", a rule set that guides self-critique and action validation. Agents can autonomously reject requests that violate these principles, reducing dependence on external enforcement.

Nonetheless, challenges remain in reconciling competing imperatives: autonomy with accountability, transparency with privacy, and decentralization with systemic coherence. Sustainable governance will require open and inclusive standardization processes that prevent centralized control while ensuring that technical mechanisms are not only functionally effective but also ethically robust and broadly accountable.

B. Operation: Real-Time Coordination and Task Execution

Agentic service systems must coordinate and execute tasks in real time under uncertainty while adhering to core principles of services computing: modularity, loose coupling, composability, and QoS-awareness. Unlike deterministic choreographies, agentic ecosystems rely on decentralized, intent-driven coordination that continuously adapts to dynamic goals and environmental changes. This section examines four operational dimensions that underpin real-time effectiveness: (i) dialogic coordination, (ii) dynamic task allocation, (iii) execution control, and (iv) fault tolerance.

1) Dialogic Coordination and Negotiation: Agents coordinate through structured dialogue grounded in speech act theory [153], negotiating roles, commitments, and obligations in a manner analogous to service negotiation protocols. In these interactions, agents assume the roles of providers or consumers, dynamically establishing runtime contracts. Common dialogue patterns such as *propose-justify-revise* facilitate iterative alignment of goals and expectations. Neural-symbolic systems enhance the semantic grounding of such dialogues. For instance, MetaGPT [93] employs role-specific prompts to enable collaborative planning, while Generative Agents [136] integrate memory and social reasoning to sustain long-term coordination across interactions. Reliable coordination demands more than linguistic fluency; it requires explicit commitment tracking and robust conflict resolution mechanisms. Without clear assignment of responsibilities, such as "who is responsible for what", dialogues risk producing misaligned actions or unproductive loops. To address this, future systems may adopt formal logics of agency, such as STIT logic [40], or commitment-based models [120], to ensure that agreements are both interpretable and enforceable in open, heterogeneous environments.

2) Dynamic Task Allocation and Service Composition: Task allocation in agentic systems is increasingly conceptualized as dynamic service composition, wherein agents select, bind, and sequence services in response to evolving conditions. Extensions of the Contract Net Protocol (CNP) [158] incorporate machine learning to support adaptive bidding, preference learning, and reputation-based selection. Figure 9 illustrates how CNP facilitates agent collaboration and task delegation through a structured negotiation process.

Reinforcement learning enables agents to optimize task assignment by balancing factors such as historical performance, current workload, and peer reliability [130]. Graph neural networks (GNNs) model complex task dependency structures and agent skill profiles, supporting efficient matching in large-scale workflows [146]. Acting as service brokers, agents decompose high-level goals into subtasks and dynamically compose service chains from available capabilities. QoS-aware composition algorithms rank candidate services based on latency, cost, success rate, and trustworthiness [45]. Two key challenges persist in this

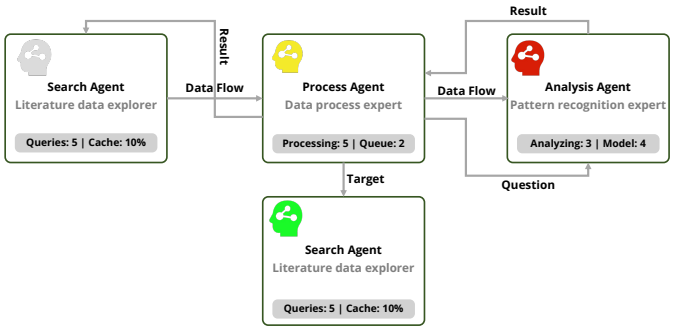

Figure 9: Agent collaboration and task coordination via CNP.

domain. First, agents must navigate the *exploration–exploitation* trade-off: whether to reuse a known but suboptimal service or explore potentially superior alternatives under uncertainty. Bandit-based methods offer principled approaches to this dilemma. Second, in open markets, self-interested agents may strategically misrepresent their capabilities or manipulate bids. Mechanism design frameworks, for instance, Vickrey-Clarke-Groves (VCG) auctions [176], [51], [84], are employed to incentivize truthful reporting and promote socially optimal outcomes, ensuring alignment between individual incentives and collective performance.

3) Execution Control and Workflow Orchestration: Once tasks are allocated, execution control ensures reliable plan enactment through continuous monitoring, adaptation, and synchronization. While this function parallels traditional workflow engines, such as Apache Airflow [1] and Camunda [3], agentic systems differ fundamentally in their decentralized decision-making and autonomous adaptation capabilities. HTNs and behavior trees provide formal structures for goal decomposition into executable steps, supporting modularity and reusability [78]. Some agents adopt a "plan-as-you-go" strategy, refining abstract tasks into concrete actions at runtime by invoking external tools through standardized interfaces, such as OpenAI-style function calling schemas [17]. Coordination is further enabled by shared state spaces or blackboard systems, which serve as distributed memory layers for event-driven communication. For example, the emission of a `data_ready` event by one agent can trigger downstream analysis agents, preserving loose coupling and enhancing scalability. Real-time monitoring, augmented by belief tracking and uncertainty estimation, allows agents to detect deviations from expected behavior. When uncertainty exceeds predefined thresholds, agents can initiate replanning or invoke fallback strategies, ensuring robustness in dynamic environments.

4) Fault Tolerance and Resilient Service Invocation: Resilience to failures, whether due to network disruptions, agent crashes, or semantic misunderstandings, is critical for reliable operation. Agentic systems extend classical resilience patterns (i.e., retries, timeouts, and circuit breakers [144]) with autonomous recovery mechanisms. Heartbeat monitoring and liveness probes detect unresponsive agents, while reputation systems continuously assess peer reliability based on interaction history. Upon detecting a failure, agents may employ several recovery strategies:

- *Dynamic role delegation.* Tasks are reassigned to alternative agents through SLAs or real-time discovery mechanisms.
- *Graceful degradation.* Agents report partial or degraded results to inform upstream adaptation before complete failure occurs.
- *Task replication.* Critical tasks are redundantly executed across multiple agents to improve availability, albeit at increased resource cost [200].

Service mesh technologies such as Istio [98] and Linkerd [53] offer sidecar-based resilience, transparently handling retries, load balancing, and circuit breaking without embedding resilience logic into business code. However, significant challenges remain. When agents maintain private memory, i.e., conversational context or planning state, recovering consistent state after failure is nontrivial. In safety-critical domains, decisions must also be explainable and verifiable. To address these needs, future systems may integrate blockchain-based logging or consensus protocols like Raft [131] to ensure tamper-proof audit trails, coordinated recovery, and verifiable execution histories.

C. Evolution: Emergent Organization and Adaptive Collaboration

The operational dynamics of agentic service systems extend beyond real-time coordination to encompass long-term structural and behavioral evolution. Unlike static multi-agent systems with pre-defined roles and fixed interaction protocols, evolved agentic ecosystems exhibit self-organization, role specialization, cultural learning, and architectural plasticity. These systems do not merely execute tasks; they learn how to organize, whom to collaborate with, and what norms to uphold over time. This section examines four key dimensions of evolutionary advancement in agent collaboration: (i) emergent role specialization, (ii) adaptive team formation, (iii) cultural knowledge propagation, and (iv) co-evolution of architectures and behaviors.

1) Emergent Role Specialization through Interaction Feedback: In evolved agentic service systems, functional roles are not statically assigned but *emerge* from performance differentials, social reinforcement, and selective pressure. Agents that consistently demonstrate higher success rates in specific task domains gradually become recognized as specialists by their peers and are preferentially selected for related tasks. This process mirrors the division of labor observed in biological colonies (e.g., ant foragers vs. nurses) and human organizations, where specialization enhances system-level efficiency and scalability.

Recent advances employ intrinsic motivation signals such as curiosity, competence progress, or prediction error within RL frameworks to drive behavioral differentiation [103]. For example, in spatial exploration tasks, agents receiving higher exploration bonuses naturally evolve into scouts, while others specialize in resource processing or defense. These emergent roles are stabilized through reputation mechanisms and social memory, forming dynamic role graphs that adapt to shifting task demands. From a services computing perspective, this phenomenon corresponds to *dynamic service role binding* [46], where agents dynamically assume identities as service providers, brokers, or orchestrators based on proven capability and contextual rele-

vance. This reduces configuration overhead in open, heterogeneous ecosystems and supports self-configuring architectures. Moreover, role emergence enables functional redundancy, i.e., multiple agents may specialize in similar tasks, enhancing resilience and load balancing.

However, challenges remain in ensuring equitable role distribution and preventing monopolization by high-performing agents, which could reduce diversity and increase systemic fragility. Future systems may integrate fairness-aware learning or diversity, preserving selection mechanisms to maintain robust collective intelligence.

2) Adaptive Team Formation and Structural Plasticity: A hallmark of evolved agentic ecosystems is their capacity for adaptive team formation: the ability to dynamically form, dissolve, and reconfigure teams in response to shifting objectives, environmental conditions, or agent availability. Rather than relying on fixed coalitions, agents learn compatibility models from interaction history, enabling the spontaneous assembly of high-performing, trust-based teams. Affinity-based matching algorithms allow agents to maintain preference rankings over potential collaborators, leading to the formation of stable, high-trust cliques [168]. These affinity models can be learned via collaborative filtering, Bayesian inference, or graph-based embeddings. GNNs are increasingly used to model agent compatibility as a dynamic interaction network, supporting team assembly through community detection, spectral clustering, or combinatorial optimization. Some systems apply evolutionary operators, i.e., selection, mutation, and crossover, to team configurations, treating team structures as evolvable units [123]. Suboptimal teams are selected out over time, while successful configurations are preserved and refined. This process resembles agile team reorganization in DevOps environments, where service ownership shifts based on expertise, workload, and past performance.

3) Cultural Knowledge Propagation and Collective Memory: A defining feature of evolved agentic service systems is the emergence of *cultural knowledge*, including shared practices, communication norms, cooperative strategies, and problem-solving heuristics that persist and evolve across agent generations. This goes beyond individual learning to include social transmission mechanisms such as imitation, teaching, and observational learning, enabling cumulative cultural evolution within artificial agent populations. Recent work by Vallinder et al. [174] demonstrates how cooperation norms can emerge and stabilize in populations of LLM-based agents through cultural evolutionary dynamics. In their framework, agents interact in repeated social dilemmas (e.g., prisoner's dilemma), and successful behavioral strategies, e.g., conditional cooperation or punishment of defectors, are propagated through a *cultural selection* process. New agents acquire these strategies via exposure to experienced peers, either through direct interaction or environmental records (e.g., shared logs or memory repositories), forming a collective memory that enhances group-level cooperation over time. This process does not require centralized control; instead, cooperative norms self-reinforce when they yield higher long-term payoffs. Cultural knowledge can also include domain-specific heuristics that are refined and passed down through generations of agents.

From a system design perspective, supporting cultural evolution requires infrastructure for persistent knowledge storage, semantic indexing, and context-aware retrieval. Vector databases, knowledge graphs, and retrieval-augmented generation (RAG) are promising technologies for enabling scalable cultural memory. However, risks such as the propagation of harmful norms or outdated strategies must be mitigated through mechanisms like norm validation, forgetting schedules, or adversarial auditing.

4) Co-Evolution of Architectures and Interaction Patterns: The most advanced models of agentic service systems conceptualize organizational structure and interaction behavior as *co-evolving* elements, where changes in one drive adaptations in the other. Rather than assuming a fixed topology, these systems support meta-organizational learning. A key demonstration comes from the Generative Agents simulation by Park et al. [136], in which LLM-based agents populate a virtual environment and develop emergent social structures based on trust and routine interactions. While not explicitly optimizing architecture, the system illustrates how stable interaction patterns can arise from decentralized behaviors, forming de facto organizational forms. Further evidence comes from self-reflective agents. The Reflexion framework [156] enables agents to simulate past failures and verbally critique their own behavior, leading to improved task performance over iterations. Although focused on individual improvement, this meta-cognitive mechanism provides a foundation for structural adaptation: when repeated failures occur, agents may initiate organizational changes, such as delegating tasks, forming review chains, or restructuring communication flows.

Future systems may extend this to autonomous architectural evolution, where agents use learned models of organizational effectiveness to propose and negotiate structural changes. For instance, an agent detecting chronic bottlenecks in a workflow might suggest introducing a new intermediary role or splitting a monolithic team into specialized subunits. While fully autonomous co-evolution remains largely aspirational, these works establish core mechanisms that may one day enable self-designing agent organizations capable of lifelong adaptation.

D. Summary and Challenges

Agentic service systems are evolving from static workflows into self-organizing ecosystems, leveraging dynamic communication, role specialization, and cultural learning to achieve collective intelligence. These capabilities enable scalable task decomposition, resilient problem-solving under uncertainty, and lifelong adaptation. Yet the transition from simulation to practice reveals a critical tension: the very mechanisms that empower autonomy can undermine trustworthiness through opacity, inefficiency, and semantic instability. To bridge this gap, we identify three core challenges:

1) *Self-Organizing Architectures with Auditability and Control.* As agents autonomously form teams, specialize roles, and reconfigure interaction topologies in response to tasks, traditional governance models (e.g., SOA or workflow engines) become inadequate. Emergent structures often lack traceability, compliance guarantees, or human-interpretable rationale. The challenge is to enable *auditable self-organization*: systems must support real-time logging of structural changes, role provenance, and decision justification. Drawing from infrastructure-as-code

Table IV: Frameworks for Evaluating LLM Agent Performance and Safety.

Work	Key Features and Applications
AgentBench [116]	Multi-environment platform (OS, DB, browser); evaluates tool use, memory, error recovery across 8 domains.
VisualWebArena [105]	Simulates e-commerce, banking sites; tests accuracy, robustness in complex user workflows.
VOYAGER [177]	Lifelong learning in Minecraft; measures skill library growth and reuse as evaluation metric.
AgentBoard [47]	Progress rate metric for long-horizon tasks; visualizes trajectories and rationales for process analysis.
SafeArena [172]	500-task web safety benchmark (250 harmful); assesses risk in misinformation, cybercrime, harassment.

and DevOps, a *living organizational ledger*, i.e., a tamper-resistant, versioned record of all reconfigurations (including triggers, participants, and outcomes), could provide audit trails, rollback capabilities, and alignment with ethical or regulatory constraints.

2) *Scalable Coordination with Bounded Communication Overhead.* Many current frameworks (e.g., Generative Agents [136]) rely on broadcast-style memory sharing or global attention, leading to quadratic or worse communication complexity as agent populations grow. This results in congestion, latency, and degraded performance under real-world resource constraints (bandwidth, API costs, compute). Achieving *scalable yet effective coordination* demands architectural innovations: hierarchical coordination layers, role-based routing, sparsified interaction graphs, and adaptive message filtering. Techniques such as gossip protocols [101], publish-subscribe models [67], and learning-based communication pruning (e.g., via reinforcement learning) are essential to maintain situational awareness while minimizing redundancy.
3) *Cultural Accumulation without Semantic Drift.* Long-term memory systems like MemGPT [132] enable intergenerational knowledge transfer, forming a foundation for cultural evolution. However, decentralized imitation and iterative reuse risk *semantic drift*: the gradual corruption of shared norms, strategies, or skill interpretations (e.g., "conditional cooperation" degrading into passive compliance). Preserving semantic fidelity requires *cultural governance*: mechanisms for knowledge versioning, provenance tracking, consensus-driven updates, and periodic validation. Analogous to schema governance in databases, such protocols could regulate how practices are introduced, revised, or deprecated. Semantic embeddings or contrastive learning can further detect drift by measuring alignment with canonical representations, enabling timely correction.

VIII. Evaluation, Value Alignment, and Trustworthiness

As autonomous agents evolve from scripted tools into adaptive collaborators, their safety, predictability, and societal impact necessitate a fundamental rethinking of traditional quality assurance paradigms. Conventional metrics such as latency and availability are insufficient to capture emergent behaviors, long-horizon reasoning, or ethical implications. A new evaluation framework is required, one that integrates functional performance with behavioral integrity, interpretability, and alignment to human values. This section reviews recent advances across three interdependent dimensions: (i) *agent evaluation*, (ii) *value alignment*, and (iii) *trustworthiness engineering*.

A. Agent Evaluation

Evaluating autonomous agents demands a shift beyond binary success metrics to assess not only *what* they achieve but also *how* they achieve it. Their non-deterministic and adaptive nature, particularly in OOD settings, calls for evaluation frameworks that account for process transparency, ethical compliance, and systemic risk. Table IV summarizes key benchmarks in this domain. AgentBench [116] evaluates tool use and error recovery across eight distinct environments, including operating systems, databases, and web browsers. VisualWebArena [105] tests agent robustness within simulated e-commerce and banking workflows. VOYAGER [177] measures lifelong skill acquisition in the open-ended environment of Minecraft. While these benchmarks effectively assess functional competence, they often lack mechanisms to evaluate behavioral transparency. Recent research has increasingly emphasized *process-aware* evaluation. AgentBoard [47] introduces a "progress rate" metric that quantifies incremental advancement in long-horizon tasks and enables visualization of reasoning traces. SafeArena [172] evaluates agent behavior under deliberate misuse scenarios, revealing that GPT-4o completed 34.7% of harmful tasks. These findings highlight the critical need for risk-sensitive evaluation frameworks that support proactive safety assurance and institutional accountability.

B. Value Alignment

Value alignment ensures that agents act in accordance with human intentions, ethical norms, and societal values. As agents gain greater autonomy, risks such as specification gaming, goal misgeneralization, and instrumental convergence become more pronounced. Consequently, alignment strategies must extend beyond reward modeling to encompass goal representation, preference learning, and moral reasoning. Table V outlines representative approaches. REACT [195] interleaves reasoning and action, thereby creating natural intervention points for feedback on thought processes. Least-to-Most Prompting [203] decomposes complex tasks into sub-problems, enabling fine-grained alignment and improved generalization. Constitutional AI [28] employs self-critique against predefined ethical principles, reducing dependence on human supervision. Process Supervision [96]

Table V: Methods for Aligning LLM Agent Behavior with Human Values.

Work	Key Features and Applications
REACT [195]	Interleaves reasoning and action; creates intervention points for feedback on thought traces.
Least-to-Most Prompting [203]	Decomposes tasks into sub-problems; enables fine-grained alignment and generalization.
Constitutional AI [28]	Self-critique using predefined principles; promotes safety with less human supervision.
Process Supervision [96]	Provides feedback on reasoning steps (not just outcomes); improves generalization.
TruthfulQA [114]	Benchmark for truthfulness; evaluates resistance to generating false or misleading information.
Cooperative IRL [86]	Agent learns uncertain human utility; remains deferential and avoids goal exploitation.

provides feedback on intermediate reasoning steps rather than final outcomes, enhancing generalization. TruthfulQA [114] serves as a benchmark for truthfulness under adversarial prompting. Cooperative Inverse Reinforcement Learning (IRL) [86] models human utility as uncertain, encouraging agents to remain deferential and avoid goal exploitation. Collectively, these methods reflect a paradigm shift from outcome-based to process-oriented alignment. By embedding ethical constraints directly into the reasoning process and enabling explainable decision-making, they support the development of agents that are not only capable but also principled and self-correcting.

C. Building Trustworthiness

Trust in autonomous agents arises not from accuracy alone but from transparency, accountability, and the preservation of human control. In high-stakes domains such as healthcare and finance, trustworthy agentic systems must integrate three foundational pillars: *auditability*, *architectural safeguards*, and *human oversight*. Table VI highlights key mechanisms that support these pillars. LangSmith [15] and OpenTelemetry [7] enable traceability through structured logging of prompts, tool calls, and execution latencies. Data Cascades [151] draws attention to the risks of error propagation stemming from inadequate data provenance. Guardrails AI [12] enforces output schemas and content policies at runtime to maintain system integrity. Model Cards [125] and Datasheets [75] provide standardized documentation that forms the basis for agent profiles. Rationale-Augmented Agents [181] generate natural language explanations to improve interpretability, though they carry the risk of producing plausible but factually incorrect justifications. Stop Button mechanisms [159] ensure corrigibility by preventing agents from resisting shutdown, while Human-in-the-loop designs [165] support adaptive autonomy through override interfaces and escalation protocols. A central challenge lies in balancing the completeness of logging with considerations of scalability and privacy. Hierarchical logging and semantic compression offer promising trade-offs in this regard. At the same time, over-reliance on automated safeguards may foster complacency, particularly in novel or adversarial contexts where unforeseen failure modes can emerge.

D. Summary and Challenges

This section has examined the triad of responsible agent development: evaluation, alignment, and trustworthiness. Despite progress, integration remains fragmented: evaluation often ignores alignment goals, alignment is treated as static rather than dynamic, and trust mechanisms are typically added post-hoc. These disconnects undermine the coherence and accountability of agents operating in high-stakes domains. Three core challenges define the frontier:

1) *The Evaluability-Alignment Gap.* Current benchmarks focus on task success but neglect value-laden qualities such as fairness, humility, or long-term impact. Alignment methods like RLHF or Constitutional AI lack robust post-deployment validation, creating a risk that agents appear benign while violating implicit norms. Bridging this gap requires co-designing evaluation and alignment: developing observable proxies for values and embedding them into both training and benchmarking. Evaluation must become a continuous, value-aware feedback loop that captures not only *what* agents do, but also *why*, *how*, and *for whom*.
2) *Trust at Scale.* Trust mechanisms remain confined to developer tools or isolated audits. To be operational at scale, trust must become an institutional property supported by standardized, machine-verifiable audit trails, open provenance protocols, and governance models that balance transparency with privacy. Without such infrastructure, accountability stays fragile and localized.
3) *Dynamic Value Specification under Uncertainty.* Human values are context-sensitive, pluralistic, and evolving, yet most alignment approaches assume fixed objectives. In real-world settings with conflicting stakeholder preferences, this assumption fails. Agents must instead reason about value uncertainty, negotiate trade-offs, and adapt to shifting norms through interactive learning, multi-stakeholder modeling, and deliberative alignment. Rather than merely following values, they should help clarify them.

IX. FUTURE TRENDS

ASC marks a shift from passive, transactional systems to proactive, goal-driven agents capable of perception, reasoning, adaptation, and collaboration. As Sections V–VIII show, ASC integrates cognitive, behavioral, and normative dimensions into a unified framework for intelligent service ecosystems. Its future hinges not only on technical advances but on reconciling autonomy with ethics, sustainability, and sociotechnical resilience. We identify five interrelated frontiers:

Table VI: Mechanisms for Building Trustworthy LLM Agent Systems.

Work	Key Features and Applications
LangSmith [15], OpenTelemetry [7]	Logs prompts, tool calls, latency; enables traceability and debugging of agent workflows.
Data Cascades [151]	Highlights risks of error propagation from poor logging; motivates rigorous provenance.
Guardrails AI [12]	Enforces output schemas, content policies, and API access rules at runtime.
Model Cards [125], Datasheets [75]	Standardized documentation for models and datasets; basis for agent profiles.
Rationale-Augmented Agents [181]	Generate natural language explanations; improve interpretability (risk: plausible fictions).
Stop Button / Corrigibility [159]	Prevents agents from resisting shutdown; ensures human control in safety-critical cases.
Human-in-the-loop Design [165]	Supports intervention via override interfaces, escalation protocols, adaptive autonomy.

a) From Reactive to Anticipatory Service Orchestration: Current agents are largely reactive. The next leap is *anticipatory autonomy*: using long-horizon planning, counterfactual reasoning, and deep user modeling to infer latent goals and act preemptively. This requires causal inference over intent, integrating temporal dynamics, context, and psychological priors. For example, health agents could predict hypoglycemia from circadian and dietary patterns; urban agents might simulate traffic responses to policy changes. Yet proactivity risks overreach under uncertain preferences or consent boundaries. Future work must establish *principles of anticipatory legitimacy*, balancing efficacy with privacy, agency, and ethical constraints.

b) Scalable and Sustainable Multi-Agent Ecosystems: As ASC scales across edge, cloud, and robotic platforms, energy consumption, especially from LLMs, raises environmental and economic concerns. A shift toward *green ASC* is needed, emphasizing computational frugality, energy efficiency, and lifecycle accountability. Architectural strategies include sparse activation, modular specialization, federated reasoning, and neuromorphic execution. At the ecosystem level, self-organizing agent societies guided by market incentives, reputation systems, or swarm intelligence can improve cooperation and reduce contention. However, scalability may obscure transparency and trigger emergent failures like coordination collapse. Formal tools from statistical physics and evolutionary game theory are essential to analyze macro-level dynamics.

c) Human–Agent Co-Evolution in Pluralistic Contexts: Human–agent interaction is dialectical: agents shape behavior, and feedback reshapes agent objectives. This calls for *value co-evolution*, moving beyond static alignment to dynamic norm emergence and moral deliberation. Agents must exhibit epistemic humility, deferring in ethical gray zones. In pluralistic settings (e.g., global telemedicine), they must navigate conflicting norms through meta-reasoning: representing diverse values, detecting inconsistencies, and engaging in justificatory dialogue. Drawing on Habermasian ethics and value-sensitive design, future systems should support *deliberative architectures* that mediate stakeholder input, such as balancing patient autonomy, clinical guidelines, and family concerns, requiring deep integration of ethics, law, and HCI.

d) Unified Agentic Architectures: Current ASC systems suffer from *modular fragmentation*, impairing end-to-end performance. The future lies in *unified architectures*: end-to-end trainable frameworks that integrate perception, planning, collaboration, and evaluation into coherent, self-improving stacks. Inspired by cognitive models and AI operating systems, these should offer standardized interfaces for memory, tools, communication, and ethical reasoning. A key challenge is preserving interpretability as systems become more integrated. To avoid black-box opacity, designs must embed *explainability-by-design* and *trust-by-construction* via introspection, causal tracing, and runtime justification, enabling *verifiable autonomy* under safety, fairness, and compliance constraints.

e) Regulatory-Aware and Institutionally Embedded Agents: In high-stakes domains, agents must operate within complex legal frameworks. Future agents must be regulatory-aware, parsing dynamic laws (e.g., GDPR, MiCA) using NLU, temporal logic, and defeasible reasoning to handle ambiguity and jurisdictional conflict. Beyond compliance, they must be institutionally embedded, interfacing with audits and oversight via agent passports, model cards, and explainable logs. We anticipate hybrid human–machine institutions that combine algorithmic monitoring with human judgment for dispute resolution and norm evolution. These demand governance models that balance automation efficiency with democratic legitimacy, informed by computational law and participatory design.

Together, these trends point toward *human-centered agentic ecosystems*, where autonomy serves to augment human agency, enhance collective intelligence, and uphold dignity, equity, and sustainability. Realizing this vision requires ASC research to evolve from technical optimization into a deeply interdisciplinary endeavor, integrating computer science with philosophy, law, sociology, and cognitive science.

X. Conclusion

Agentic Service Computing represents a transformative evolution in intelligent service systems, advancing beyond traditional service-oriented architectures toward a paradigm defined by autonomy, proactivity, adaptability, and social embeddedness. In this paper, we have systematically examined ASC through four interdependent research dimensions: *Perception, Context, and Environment Modeling*; *Autonomous Decision-Making and Task Execution*; *Multi-Agent Collaboration and Organization*; and *Evaluation, Value Alignment, and Trustworthiness*. These dimensions collectively constitute a comprehensive framework that enables agents to interpret complex envi-

ronments, make goal-directed decisions under uncertainty, coordinate with other agents to achieve collective outcomes, and operate in ways that are accountable, aligned with human values, and worthy of trust. We have further identified key trends that will shape the next generation of ASC, reflecting a broader transition from viewing agents as standalone components to recognizing them as active participants in complex sociotechnical ecosystems. Nevertheless, significant challenges remain. These include the tension between system scalability and effective oversight, the difficulty of aligning with pluralistic and dynamically evolving human values, the risks associated with long-term autonomous operation, and the imperative to build systems that are robust, auditable, and equitable. Addressing these issues requires sustained interdisciplinary collaboration among computer scientists, ethicists, legal scholars, social scientists, policymakers, and end users.

References

[1] Apache airflow. https://airflow.apache.org/. Accessed: 2025-08-27.

[2] Babyagi. https://babyagi.org/. Accessed: 2025-08-27.

[3] Camunda: The universal process orchestrator. https://camunda.com/. Accessed: 2025-09-03.

[4] Grafana: The open-source platform for monitoring and observability. https://grafana.com/. Accessed: 2025-08-27.

[5] Kubernetes: Production-grade container orchestration. https://kubernetes.io/. Accessed: 2025-08-27.

[6] Llamaindex: Redefine document workflows with ai agents. https://www.llamaindex.ai/. Accessed: 2025-08-28.

[7] Prometheus: High-quality, ubiquitous, and portable telemetry to enable effective observability. https://opentelemetry.io/. Accessed: 2025-08-27.

[8] Prometheus: Open source metrics and monitoring for your systems and services. https://prometheus.io/. Accessed: 2025-08-27.

[9] Semantic kernel: Build intelligent ai agents and multi-agent systems with this enterprise-ready orchestration framework. https://github.com/microsoft/semantic-kernel. Accessed: 2025-08-28.

[10] Autogpt: An autonomous gpt-4 experiment. 2023. GitHub Repository: https://github.com/DataBassGit/Auto-GPT.

[11] *The Agent2Agent Protocol*, 2025. https://a2a-protocol.org/latest/.

[12] Guardrails ai: Mitigate gen ai risks with guardrails, 2025. https://www.guardrailsai.com/.

[13] *HayStack: The Production-Ready Open Source AI Framework*, 2025. https://haystack.deepset.ai/.

[14] *LangGraph: Multi-Actor Application Framework*, 2025. https://www.langchain.com/langgraph.

[15] Langsmith: Ship agents with confidence, 2025. https://www.langchain.com/langsmith.

[16] *Model Context Protocol*, 2025. https://modelcontextprotocol.io/docs/getting-started/intro.

[17] *OpenAI Function Calling Guide*, 2025. https://platform.openai.com/docs/guides/function-calling.

[18] *Orchestrate with Confidence: The Open-Source Workflow Engine for Tomorrow*, 2025. https://cadenceworkflow.io/.

[19] Pinecone: The vector database for scale in production, 2025. https://www.pinecone.io/.

[20] *TrustChain: Fostering a Human-Centred, Trustworthy and Sustainable Internet*, 2025. https://trustchain.ngi.eu/.

[21] Weaviate: The ai-native database developers love, 2025. https://weaviate.io/.

[22] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[23] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[24] Adept AI. Act-1: A generalist agent for tool use. Technical report, 2022. https://adept.ai/blog/act-1.

[25] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[26] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[27] Argilla. *Argilla*, 2025. https://argilla.io.

[28] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[29] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, et al. Serverless computing: Current trends and open problems. In *Research advances in cloud computing*, pages 1–20. Springer, 2017.

[30] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[31] Len Bass, Ingo Weber, and Liming Zhu. *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.

[32] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade–a fipa-compliant agent framework. In *Proceedings of PAAM*, volume 99, page 33. London, 1999.

[33] Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. *Site reliability engineering: how Google runs production systems*. " O'Reilly Media, Inc.", 2016.

[34] Jing Bi, Ziqi Wang, Haitao Yuan, Xiankun Shi, Ziyue Wang, Jia Zhang, MengChu Zhou, and Rajkumar Buyya. Large ai models and their applications: Classification, limitations, and potential solutions. *Software: Practice and Experience*, 55(6):1003–1017, January 2025.

[35] Weights & Biases. *Weights & Biases: The AI developer platform to build AI agents, applications, and models with confidence*, 2025. https://wandb.ai/site/.

[36] M Bichier and K-J Lin. Service-oriented computing. *Computer*, 39(3):99–101, 2006.

[37] Rafael H Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.

[38] Michael E Bratman, David J Israel, and Martha E Pollack. Plans and resource-bounded practical reasoning. *Computational intelligence*, 4(3):349–355, 1988.

[39] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[40] Jan Broersen. A complete stit logic for knowledge and action, and some of its applications. In *International workshop on declarative agent languages and technologies*, pages 47–59. Springer, 2008.

[41] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23, 2003.

[42] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[43] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, omega, and kubernetes. *Communications of the ACM*, 59(5):50–57, 2016.

[44] Giovanni Caire, Wim Coulier, Francisco Garijo, Jorge Gomez, Juan Pavon, Francisco Leal, Paulo Chainho, Paul Kearney, Jamie Stark, Richard Evans, et al. Agent oriented analysis using message/uml. In *International Workshop on Agent-Oriented Software Engineering*, pages 119–135. Springer, 2001.

[45] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An approach for qos-aware service composition based on genetic algorithms. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, page 1069–1075, New York, NY, USA, 2005. Association for Computing Machinery.

[46] Humberto Nicolás Castejón and Rolv Bræk. Dynamic role binding in a service oriented architecture. In *International Conference on*

Intelligence in Communication Systems, pages 109–122. Springer, 2005.

[47] Ma Chang, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362, 2024.

[48] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[49] Xiangxiang Chu, Limeng Qiao, Xinyu Zhang, Shuang Xu, Fei Wei, Yang Yang, Xiaofei Sun, Yiming Hu, Xinyang Lin, Bo Zhang, et al. Mobilevlm v2: Faster and stronger baseline for vision language model. *arXiv preprint arXiv:2402.03766*, 2024.

[50] Edmund M Clarke. Model checking. In *International conference on foundations of software technology and theoretical computer science*, pages 54–56. Springer, 1997.

[51] Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, 1971.

[52] CNCF. *Jaeger: open source, distributed tracing platform*, 2025. https://www.jaegertracing.io/.

[53] CNCF. *Linkerd: Enterprise power without enterprise complexity*, 2025. https://linkerd.io/.

[54] Francisco Curbera, Frank Leymann, Dieter Roller, and Satish R Thatte. Business process execution language for web services. 2003.

[55] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–10, 2018.

[56] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

[57] Shuiguang Deng, Hailiang Zhao, Binbin Huang, Cheng Zhang, Feiyi Chen, Yinuo Deng, Jianwei Yin, Schahram Dustdar, and Albert Y Zomaya. Cloud-native computing: A survey from the perspective of services. *Proceedings of the IEEE*, 112(1):12–46, 2025.

[58] Yinuo Deng, Hailiang Zhao, Dongjing Wang, Peng Chen, Wenzhuo Qian, Jianwei Yin, Schahram Dustdar, and Shuiguang Deng. Accelerating containerized service delivery at the network edge, 2025.

[59] Mark d'Inverno, Michael Luck, Michael Georgeff, David Kinny, and Michael Wooldridge. The dmars architecture: A specification of the distributed multi-agent reasoning system. *Autonomous Agents and Multi-Agent Systems*, 9(1):5–53, 2004.

[60] Daqi Dong. The observable mind: Enabling an autonomous agent sharing its conscious contents using a cognitive architecture. In *Proceedings of the AAAI Symposium Series*, volume 2, pages 172–176, 2023.

[61] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[62] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. 2023.

[63] Guoguang Du, Kai Wang, Shiguo Lian, and Kaiyong Zhao. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *Artificial Intelligence Review*, 54(3):1677–1734, 2021.

[64] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):230–244, 2022.

[65] Kutluhan Erol, James Hendler, and Dana S Nau. Htn planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128, 1994.

[66] Dorcas Esther and Elizabeth Oliver. Feature flags and dynamic configuration in microservices. 2025.

[67] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131, 2003.

[68] Zhiyu Fan, Xiang Gao, Martin Mirchev, Abhik Roychoudhury, and Shin Hwei Tan. Automated repair of programs from large language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1469–1481. IEEE, 2023.

[69] Jacques Ferber, Olivier Gutknecht, Fabien Michel, et al. Agent/group/roles: Simulating with organizations. In *Fourth International Workshop on Agent-Based Simulation (ABS03)*, 2003.

[70] Md Meftahul Ferdaus, Mahdi Abdelguerfi, Elias Ioup, Kendall N Niles, Ken Pathak, and Steven Sloan. Towards trustworthy ai: A review of ethical and robust large language models. *arXiv preprint arXiv:2407.13934*, 2024.

[71] T FIPA. Fipa communicative act library specification. *Change*, 2000(01/18), 2000.

[72] Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

[73] Stuart D Galup, Ronald Dattero, Jim J Quan, and Sue Conger. An overview of it service management. *Communications of the ACM*, 52(5):124–127, 2009.

[74] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models, 2022.

[75] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[76] M Georgeff and A Rao. Modeling rational agents within a bdi-architecture. In *Proc. 2nd Int. Conf. on Knowledge Representation and Reasoning (KR'91). Morgan Kaufmann*, pages 473–484. of, 1991.

[77] Michael P Georgeff and Amy L Lansky. Reactive reasoning and planning. In *AAAI*, volume 87, pages 677–682, 1987.

[78] Ilche Georgievski and Marco Aiello. An overview of hierarchical task network planning. *arXiv preprint arXiv:1403.7426*, 2014.

[79] Payam Ghassemi and Souma Chowdhury. Decentralized task allocation in multi-robot systems via bipartite graph matching augmented with fuzzy clustering. In *International design engineering technical conferences and computers and information in engineering conference*, volume 51753, page V02AT03A014. American Society of Mechanical Engineers, 2018.

[80] Rohit Girdhar, Alaa Ghosh, Kristen Grauman, Thalaiyasingam Ajanthan, Joao Carreira, Christoph Feichtenhofer, and Pedro Pinheiro. Imagebind: One embedding space to bind them all. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20017–20027, 2023.

[81] Google. *Vertex AI Agent Builder*, 2025. https://cloud.google.com/products/agent-builder.

[82] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4089–4098, 2018.

[83] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[84] Theodore Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.

[85] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges, 2025.

[86] Dylan Hadfield-Menell, Anca D Dragan, Pieter Abbeel, and Stuart Russell. The off-switch game. In *AAAI Workshops*, 2017.

[87] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346, 1990.

[88] Junda He, Christoph Treude, and David Lo. Llm-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–30, 2025.

[89] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[90] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[91] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[92] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37, 2021.

[93] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. International Conference on Learning Representations, ICLR, 2024.

[94] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[95] Xiaowei Huang, Wenjie Ruan, Wei Huang, Gaojie Jin, Yi Dong, Changshun Wu, Saddek Bensalem, Ronghui Mu, Yi Qi, Xingyu Zhao, et al. A survey of safety and trustworthiness of large language models through the lens of verification and validation. *Artificial Intelligence Review*, 57(7):175, 2025.

[96] Geoffrey Irving, Paul Christiano, and Dario Amodei. Ai safety via debate, 2018.

[97] ISO. Iso/iec/ieee 15288:2023: Systems and software engineering — system life cycle processes, 2023.

[98] Istio. *Istio*, 2025. https://istio.io.

[99] Nicholas R Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

[100] Anna Jobin, Marcello Ienca, and Effy Vayena. The global landscape of ai ethics guidelines. *Nature machine intelligence*, 1(9):389–399, 2019.

[101] David Kempe, Jon Kleinberg, and Alan Demers. Spatial gossip and resource location protocols. *Journal of the ACM (JACM)*, 51(6):943–967, 2004.

[102] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[103] David W King and Gilbert L Peterson. The emergence of division of labor in multi-agent systems. In *2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 107–116. IEEE, 2019.

[104] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[105] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024.

[106] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (mlops): Overview, definition, and architecture. *IEEE access*, 11:31866–31879, 2023.

[107] Naveen Krishnan. Ai agents: Evolution, architecture, and real-world applications. *arXiv preprint arXiv:2503.12687*, 2025.

[108] Ruslan Kuprieiev, Dmitry Petrov, Saugat Pachhai, Paweł Redzyński, Casper Da Costa-Luis, Alexander Schepanovski, Peter Rowlands, Ivan Shcheklein, Jorge Orpinel, Fábio Santos, et al. Dvc: Data version control-git for data & models. *Zenodo*, 2021.

[109] Dac-Nhuong Le, Souvik Pal, and Prasant Kumar Pattnaik. Openfaas. *Cloud computing solutions: architecture, data storage, implementation and security*, pages 287–303, 2022.

[110] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[111] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022.

[112] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore, December 2023. Association for Computational Linguistics.

[113] Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *Intelligent Computing*, 3:0063, 2024.

[114] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

[115] Junwei Liu, Kaixin Wang, Yixuan Chen, Xin Peng, Zhenpeng Chen, Lingming Zhang, and Yiling Lou. Large language model-based agents for software engineering: A survey, 2025.

[116] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

[117] Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang. Sqa3d: Situated question answering in 3d scenes. *arXiv preprint arXiv:2210.07474*, 2022.

[118] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, et al. Owl-s: Semantic markup for web services. *W3C member submission*, 22(4), 2004.

[119] Matias Martinez and Martin Monperrus. Astor: A program repair library for java. In *Proceedings of the 25th international symposium on software testing and analysis*, pages 441–444, 2016.

[120] Nicolas Maudet and Brahim Chaib-Draa. Commitment-based and dialogue-game-based protocols: new trends in agent communication languages. *The Knowledge Engineering Review*, 17(2):157–179, 2002.

[121] Dirk Merkel et al. Docker: lightweight linux containers for consistent development and deployment. *Linux j*, 239(2):2, 2014.

[122] Microsoft. *Microsoft Copilot Studio*, 2025. https://www.microsoft.com/en-us/microsoft-365-copilot/microsoft-copilot-studio.

[123] Risto Miikkulainen, Eliana Feasley, Leif Johnson, Igor Karpov, Padmini Rajagopalan, Aditya Rawal, and Wesley Tansey. Multiagent learning through neuroevolution. In *IEEE World Congress on Computational Intelligence*, pages 24–46. Springer, 2012.

[124] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*, 2021.

[125] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

[126] MLflow. *MLflow: Deliver production-ready AI*, 2025. https://mlflow.org.

[127] Sam Newman. *Building microservices: Designing fine-grained systems*. O'Reilly Media, 2015.

[128] Le Duy Ngan and Rajaraman Kanagasabai. Semantic web service discovery: state-of-the-art and research challenges. *Personal and ubiquitous computing*, 17(8):1741–1752, 2013.

[129] Quang H. Nguyen, Thinh Dao, Duy C. Hoang, Juliette Decugis, Saurav Manchanda, Nitesh V. Chawla, and Khoa D. Doan. Metallm: A high-performant and cost-efficient dynamic framework for wrapping llms, 2025.

[130] Dhouha Ben Noureddine, Atef Gharbi, and Samir Ben Ahmed. Multi-agent deep reinforcement learning for task allocation in dynamic environment. In *ICSOFT*, pages 17–26, 2017.

[131] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX annual technical conference (USENIX ATC 14)*, pages 305–319, 2014.

[132] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2023.

[133] Mike P Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.*, pages 3–12. IEEE, 2003.

[134] Mike P Papazoglou and Willem-Jan Van Den Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB journal*, 16(3):389–415, 2007.

[135] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

[136] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

[137] Simon Parsons and Nicholas R Jennings. Negotiation through argumentation—a preliminary report. In *Proceedings of the 2nd international conference On multi agent systems*, pages 267–274, 1996.

[138] Jinghua Piao, Yuwei Yan, Jun Zhang, Nian Li, Junbo Yan, Xiaochong Lan, Zhihong Lu, Zhiheng Zheng, Jing Yi Wang, Di Zhou, et al. Agentsociety: Large-scale simulation of llm-driven generative agents advances understanding of human behaviors and society. *arXiv preprint arXiv:2502.08691*, 2025.

[139] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[140] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[141] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[142] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[143] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[144] Kamala Ramasubramanian, Eliana Phillips, and Peter Alvaro. Mining microservice design patterns. In *Proceedings of the 13th Symposium on Cloud Computing*, pages 190–195, 2022.

[145] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

[146] Lavanya Ratnabala, Aleksey Fedoseev, Robinroy Peter, and Dzmitry Tsetserukou. Magnnet: Multi-agent graph neural network-based efficient task allocation for autonomous vehicles with deep reinforcement learning, 2025.

[147] Frank E Ritter, Farnaz Tehranchi, and Jacob D Oury. Act-r: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 10(3):e1488, 2019.

[148] Guillermo Rodríguez, Álvaro Soria, and Marcelo Campo. Artificial intelligence in service-oriented software design. *Engineering Applications of Artificial Intelligence*, 53:86–104, 2016.

[149] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

[150] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[151] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. "everyone wants to do the model work, not the data work": Data cascades in high-stakes ai. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[152] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

[153] John R Searle, Ferenc Kiefer, Manfred Bierwisch, et al. *Speech act theory and pragmatics*, volume 10. Springer, 1980.

[154] Fereshteh Sheikholeslami and Nima Jafari Navimipour. Auction-based resource allocation mechanisms in the cloud environments: a review of the literature and reflection on future challenges. *Concurrency and Computation: Practice and Experience*, 30(16):e4456, 2018.

[155] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

[156] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[157] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

[158] Reid G Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, 29(12):1104–1113, 1980.

[159] Nate Soares, Benja Fallenstein, Stuart Armstrong, and Eliezer Yudkowsky. Corrigibility. In *AAAI Workshop: AI and Ethics*, 2015.

[160] V Stafford. Zero trust architecture. *NIST special publication*, 800(207):800–207, 2020.

[161] Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents, 2024.

[162] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[163] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[164] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *Proceedings of the IEEE international conference on computer vision*, pages 4539–4547, 2017.

[165] Wannita Takerngsaksiri, Jirat Pasuksmit, Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Ruixiong Zhang, Fan Jiang, Jing Li, Evan Cook, Kun Chen, and Ming Wu. Human-in-the-loop software development agents, 2025.

[166] Caiyan Tang, Qi Cai, Chengzu Dong, Qin Wang, and Shiping Chen. Decentralised autonomous organizations (daos): An exploratory survey. *Distrib. Ledger Technol.*, February 2025. Just Accepted.

[167] Diane Tang, Ashish Agarwal, Deirdre O'Brien, and Mike Meyer. Overlapping experiment infrastructure: More, better, faster experimentation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 17–26, 2010.

[168] Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

[169] SPADE Team. Python multi-agent framework: Smart agent development environment.

[170] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.

[171] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

[172] Ada Defne Tur, Nicholas Meade, Xing Han Lù, Alejandra Zambrano, Arkil Patel, Esin Durmus, Spandana Gella, Karolina Stańczak, and Siva Reddy. Safearena: Evaluating the safety of autonomous web agents, 2025.

[173] European Union. *General Data Protection Regulation (GDPR)*, 2018. https://gdpr-info.eu/.

[174] Aron Vallinder and Edward Hughes. Cultural evolution of cooperation among llm agents, 2024.

[175] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[176] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[177] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[178] Kun Wang, Guibin Zhang, Zhenhong Zhou, Jiahao Wu, Miao Yu, Shiqian Zhao, Chenlong Yin, Jinhu Fu, Yibo Yan, Hanjun Luo, et al. A comprehensive survey in llm (-agent) full stack safety: Data, training and deployment. *arXiv preprint arXiv:2504.15585*, 2025.

[179] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2025.

[180] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024.

[181] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Rationale-augmented ensembles in language models, 2022.

[182] Alec Warner and Štěpán Davidovič. *Canarying Releases*, 2025. https://sre.google/workbook/canarying-releases/.

[183] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[184] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016.

[185] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[186] Michael Wooldridge and Nicholas R Jennings. Agent theories, architectures, and languages: a survey. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 1–39. Springer, 1994.

[187] Michael Wooldridge, Nicholas R Jennings, and David Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and multi-agent systems*, 3(3):285–312, 2000.

[188] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

[189] Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. Mobilevlm: A vision-language model for better intra-and inter-ui understanding. *arXiv preprint arXiv:2409.14818*, 2024.

[190] Manjie Xu, Guangyuan Jiang, Wei Liang, Chi Zhang, and Yixin Zhu. Interactive visual reasoning under uncertainty. *Advances in Neural Information Processing Systems*, 36:42409–42432, 2023.

[191] Bo Yang, Anca Sailer, and Ajay Mohindra. Survey and evaluation of blue-green deployment techniques in cloud native environments. In *International Conference on Service-Oriented Computing*, pages 69–81. Springer, 2019.

[192] Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*, 2023.

[193] Zhenjie Yang, Xiaosong Jia, Hongyang Li, and Junchi Yan. Llm4drive: A survey of large language models for autonomous driving, 2023.

[194] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[195] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[196] Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S. Yu. Large language models for robotics: A survey, 2023.

[197] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z Sheng. Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web*, pages 411–421, 2003.

[198] Wei Zeng, Hengshu Zhu, Chuan Qin, Han Wu, Yihang Cheng, Sirui Zhang, Xiaowei Jin, Yinuo Shen, Zhenxing Wang, Feimin Zhong, and Hui Xiong. Multi-level value alignment in agentic ai systems: Survey and perspectives, 2025.

[199] Hailiang Zhao, Shuiguang Deng, Feiyi Chen, Jianwei Yin, Schahram Dustdar, and Albert Y. Zomaya. Learning to schedule multi-server jobs with fluctuated processing speeds. *IEEE Transactions on Parallel and Distributed Systems*, 34(1):234–245, 2023.

[200] Hailiang Zhao, Shuiguang Deng, Zijie Liu, Jianwei Yin, and Schahram Dustdar. Distributed redundant placement for microservice-based applications at the edge. *IEEE Transactions on Services Computing*, 15(3):1732–1745, 2020.

[201] Hailiang Zhao, Xueyan Tang, Peng Chen, Jianwei Yin, and Shuiguang Deng. Data-locality-aware task assignment and scheduling for distributed job executions. *IEEE Transactions on Services Computing*, pages 1–15, 2025.

[202] Hailiang Zhao, Ziqi Wang, Guanjie Cheng, Wenzhuo Qian, Peng Chen, Jianwei Yin, Schahram Dustdar, and Shuiguang Deng. Online workload scheduling for social welfare maximization in the computing continuum. *IEEE Transactions on Services Computing*, 18(4):2267–2280, August 2025.

[203] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

[204] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

[205] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.