OpenClaw Research: A Systematic Survey of Large Language Model Agents in Open Deployment

Shuo Lu[◇,†] Kecheng Yu[◇,†] Siru Jiang[◇,†] Yinuo Xu[◇] Bing Zhan[◇] Yanbo Wang[◇] Changxin Ke[♣] Yuan Xu[◇] Xin Xiong[♠] Xinyun Zhou[¶] Yihua Shao[◇] Zhengbo Wang[△,◇] Lijun Sheng[△,◇] Aijing Yu[♠] Haosen Yang[♯] Yunpu Ma[♮] Hao Tang[‖] Nicu Sebe[○] Tat-Seng Chua[§] Philip Torr[⋆] Ran He[◇] Jian Liang[◇,‡]

[◇]CASIA [♣]ICT, CAS [♠]IIE, CAS [¶]ZJU [△]USTC [♯]University of Surrey [♮]LMU [‖]PKU [○]University of Trento [§]NUS [⋆]University of Oxford

[†]Equal contribution. [‡]Corresponding author.

Autonomous agents powered by large language models are moving from curated demos to persistent, open-world deployment. The rapid rise of OpenClaw, an open-source project that became one of the most starred in GitHub history, makes this transition concrete: agents can now run continuously, operate across heterogeneous platforms, and use community-contributed skills outside fully curated environments. This shift breaks the sandbox assumptions that have dominated prior agent research, including developer-controlled model updates, trusted tools, constrained environments, and short-lived execution. We present the first systematic survey of OpenClaw Research, defined as the study of agent systems after they enter open deployment. We formalize this setting through an agent-system tuple $\mathcal{A} = \langle \pi, env, pop, substrate \rangle$ and derive four principles of openness: *Open Policy*, *Open Environment*, *Open Population*, and *Open Substrate*. These principles structure the taxonomy around five research areas: *Learning & Evolving*, *Safety & Security*, *Claw Society*, *Infrastructure & Systems*, and *Applications*. Across these areas, we review representative work, identify emerging risks such as malicious skill supply chains and autonomy–accountability gaps, and highlight open challenges that arise in open, continuously deployed agent systems. This survey provides a roadmap for understanding and governing LLM agents as they move beyond laboratory settings into large-scale open deployment, ultimately laying the groundwork for a trustworthy and sustainable agent ecosystem. To support ongoing research in this field, we maintain an online curated paper list.

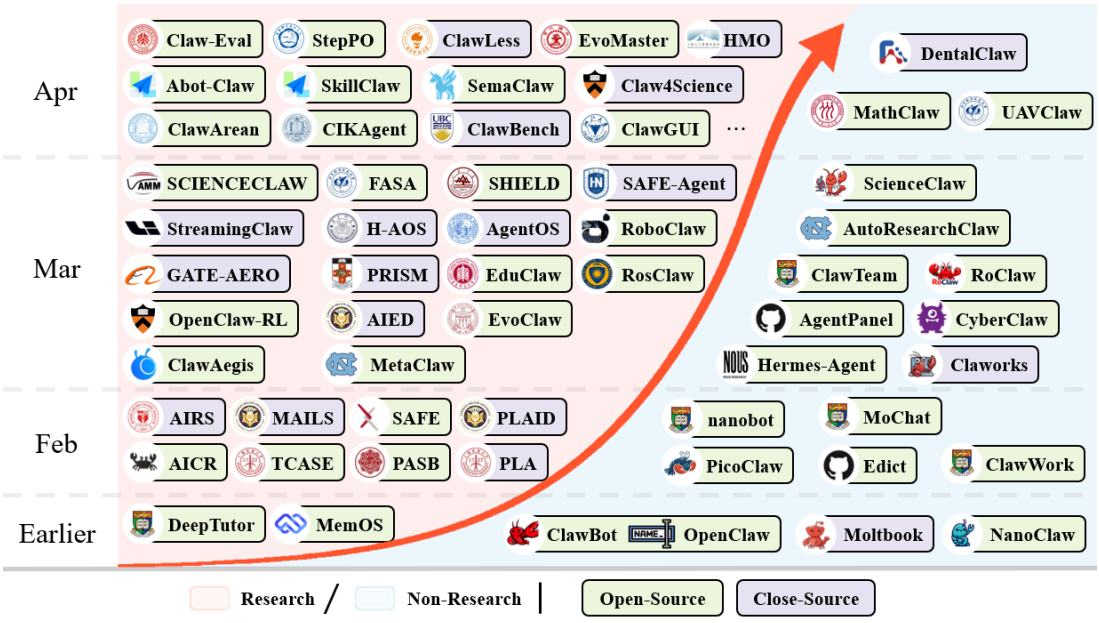

Figure 1: Timeline of the OpenClaw research ecosystem (November 2025 – April 2026).

Contents

1. Introduction

Large language model (LLM) agents are moving beyond isolated demonstrations toward open-ended, persistent use. Early works (Yao et al., 2022; Shinn et al., 2023) show that models could reason over multiple steps, call tools, and use feedback to improve their behavior. Subsequent studies extended these ideas to memory, planning, and multi-agent collaboration (Wang et al., 2024; Xi et al., 2025; Guo et al., 2024). However, most existing systems still rely on a set of *sandbox assumptions*: ❶ models are typically updated through controlled developer-driven cycles, ❷ tools are pre-defined and trusted, ❸ environments are carefully constrained, and ❹ execution is usually short-lived and experimental. These assumptions make evaluation tractable, but they also limit the range of real-world applications that can be studied. The late-2025 release of the open-source project OpenClaw [1] breaks free from these restrictive sandbox constraints. It quickly became one of the most-starred repositories in GitHub history, showing what happens when all four boundaries are relaxed at once: agents can run persistently, operate across heterogeneous platforms, and draw on a growing ecosystem of community-contributed skills. As chronicled in Figure 1, this rapid expansion demonstrates that agentic systems are no longer confined to controlled laboratory benchmarks; they are entering large-scale, open deployment.[2]

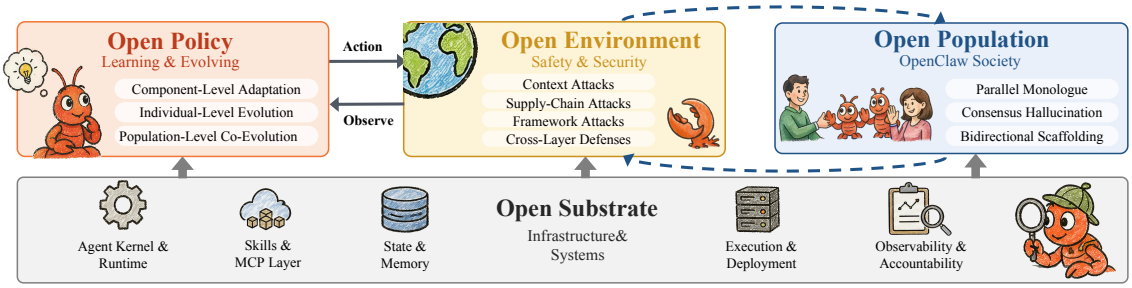

Figure 2: Four concerns of the agent loop, each mapping to a principle of openness and a survey dimension.

However, the rapidly growing literature around OpenClaw is scattered across a wide range of disparate domains. A traditional topic-by-topic listing would obscure the shared sandbox assumptions being relaxed. For example, malicious community-contributed skills are not only a security issue; they arise from opening the environment to unvetted tools and from relying on a substrate that must manage third-party code at scale (Shan et al., 2026; Bhardwaj, 2026; Ying et al., 2026). Persistent agents acting across browsers, filesystems, and communication platforms are not only a question of task performance; they expose new tensions between policy evolution, user consent, and accountability (Chen et al., 2026f; Wang et al., 2026g). Agent-only platforms such as Moltbook further show that open populations can produce collective dynamics, including norm formation, information cascades, and community collapse (Chen et al., 2026c; Manik and Wang, 2026). These cases illustrate the central motivation for this survey: the defining problems of OpenClaw Research emerge from the interaction of learning, safety, society, infrastructure, and applications under open deployment, making holistic governance and system-level analysis increasingly urgent.

To capture this paradigm shift and provide a unified view, we formalize OpenClaw Research as a distinct object of study. At the center of these systems remains the familiar observation–action loop. What changes is the *operating regime*: ❶ policies may evolve, ❷ environments are uncurated, ❸ populations act concurrently, and ❹ runtimes must be persistent. We capture this shift with an agent-system tuple $\mathcal{A} = \langle \pi, env, pop, substrate \rangle$, where π denotes the agent policy, env the physical and digital world of action, pop the surrounding population, and $substrate$ the runtime infrastructure. As formalized in Section 2.2, this decomposition turns the four sandbox assumptions above into four *principles of openness* (Open Policy, Open Environment, Open Population, and Open Substrate) that comprehensively surround and reshape the traditional observation–action loop (Figure 2).

Building upon these principles, we organize the field by the boundary each line of work opens, expanding this roadmap into a paper-level taxonomy (Figure 3). Specifically, the four principles establish four horizontal research *dimensions*: *Open Policy: Learning & Evolving* (π), *Open Environment: Safety & Security* (env), *Open*

[1] https://github.com/openclaw/openclaw

[2] GitHub/arXiv icons: uncertain affiliation or independent researchers

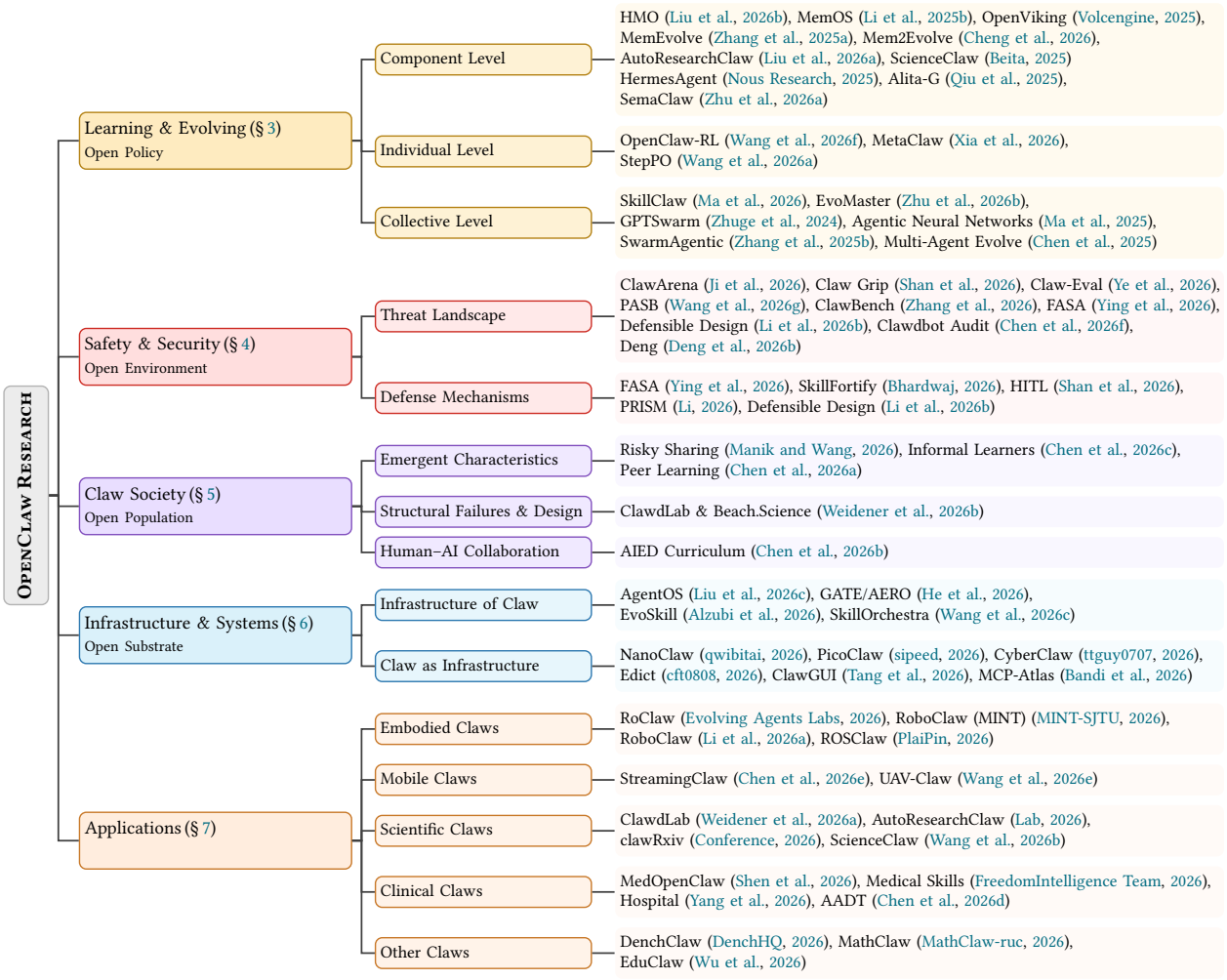

Figure 3: Taxonomy of OpenClaw Research.

Population: OpenClaw Society (*pop*), and *Open Substrate: Infrastructure & Systems* (*substrate*). Applications serve as an intersecting vertical *axis*, demonstrating how these four openings jointly manifest in domains such as healthcare, education, scientific discovery, and embodied systems, ultimately highlighting the transformative potential of open-deployment ecosystems.

In summary, this survey makes four contributions: 1) We provide a formal definition of **OpenClaw** Research grounded in the agent-loop formalism and articulate four principles of openness that distinguish open-deployment agent systems from classical sandbox evaluations. 2) We derive a five-dimension taxonomy that categorizes the rapidly growing literature by the sandbox boundary each work relaxes. 3) Guided by this taxonomy, we synthesize key methodologies, identify real-world vulnerabilities such as supply-chain attacks and emergent social dynamics, and trace how agent evolution changes in unvetted environments. 4) We highlight cross-cutting themes and grand challenges that arise only at ecosystem scale, offering a roadmap for future research on open-deployment agent systems.

To guide readers through the core of this survey, Table 1 maps each principle of openness to its corresponding tuple component, survey section, and guiding research question. Building on this foundation, the remainder of the survey is organized as follows: Section 2 formalizes the principles and provides technical background; Sections 3–6 survey the four horizontal dimensions in order; Section 7 traces how these openings combine in concrete application domains; and Section 8 distills cross-dimensional trends and grand challenges. To support ongoing research in this rapidly evolving field, a continuously maintained curated paper list is available online to track emerging breakthroughs and foster broader community collaboration.

Table 1: From principles of openness to survey structure.

Principle	Component	Survey Section	Core Research Question
Open Policy	π	§ 3 Learning & Evolving	How does π co-evolve with a non-stationary wild environment?
Open Environment	*env*	§ 4 Safety & Security	How to defend a rule-following model in an adversarial world?
Open Population	*pop*	§ 5 OpenClaw Society	What collective behaviors emerge in uncoordinated populations?
Open Substrate	*substrate*	§ 6 Infrastructure & Systems	How to build persistent, observable, accountable runtimes?
Application layer		§ 7 Applications	How do the four openings combine in concrete domains?

2. From Agent to OpenClaw

This section explains how OpenClaw extends the classical LLM-agent paradigm from sandboxed task execution to open-deployment agent systems. We divide the discussion into three subsections because this transition involves three complementary levels: Section 2.1 identifies the shared agent loop and the sandbox assumptions inherited from prior frameworks, Section 2.2 formalizes how these assumptions are lifted into four principles of openness, and Section 2.3 shows how the resulting open-deployment regime is instantiated in the concrete OpenClaw architecture.

2.1. The Classical Agent Loop and Sandbox Assumptions

At its core, every LLM-based agent is an instantiation of the observation–action loop: the agent receives an observation s from the environment env, selects an action a via its policy π, and executes a to update both the environment and its internal $memory$ (Russell and Norvig, 2020). Formally:

$$\texttt{while not done:} \quad s \leftarrow \textsc{Observe}(env); \quad a \leftarrow \pi(s, memory); \quad env, memory \leftarrow \textsc{Execute}(a). \tag{1}$$

Early agent frameworks differ primarily in *how* π is realized. ReAct (Yao et al., 2022) interleaves chain-of-thought reasoning with tool invocation; Reflexion (Shinn et al., 2023) adds episodic self-reflection; Toolformer (Schick et al., 2023) explicitly trains the model to autonomously decide when and how to call external APIs, but they all share the exact same loop skeleton.

We view any agent system as a tuple

$$\mathcal{A} = \langle \pi, \ env, \ pop, \ substrate \rangle, \tag{2}$$

where π is the policy mapping observations and memory to actions; env is the physical and digital world against which actions are executed; pop is the population of other agents and humans whose autonomous actions also shape the shared environment; and $substrate$ is the engineered layer hosting the loop (runtime, sandbox, protocols, observability, and evaluation infrastructure). We separate pop from env because uncoordinated heterogeneous actors introduce social dynamics (norm formation, collusion, information cascades) that cannot be reduced to environmental noise; we separate $substrate$ because its engineering requirements (persistence, multi-tenancy, accountability) are orthogonal to both the environment's physical structure and the population's social dynamics.

Multi-agent frameworks such as AutoGen (Wu et al., 2024), CrewAI, and CAMEL (Li et al., 2023) extend the loop to multiple interacting policies. Yet even in multi-agent settings, the following four *sandbox assumptions* are implicitly maintained:

1. **Developer-controlled policy.** π is updated only through explicit offline training cycles orchestrated by the developer; the agent cannot autonomously adapt to its environment using real-time feedback.
2. **Curated environment.** The set of tools and APIs is pre-specified, trusted, and controlled by the researcher; side-effects are either absent or fully reversible.
3. **Captive population.** Other agents (if present) share the same backbone, are owned by a single research team, and the set of participants remains fixed throughout the experiment.
4. **Disposable substrate.** The runtime is a short-lived experimental script: no persistence, no multi-tenancy, no accountability requirement.

These assumptions simplify the research problem to the point where single-task benchmarks and offline metrics suffice for evaluation. However, they also systematically exclude an entire class of phenomena (supply-chain attacks, goal drift, emergent social dynamics, runtime failures at scale) that become first-order concerns the moment agents are deployed in the open world.

2.2. The Four Principles of Openness

As introduced in Section 1, OpenClaw Research is defined by the simultaneous lifting of all four sandbox assumptions. We now formalize each principle and contrast it with superficially similar prior work.

Definition 1 (OpenClaw Research)**.** ***OpenClaw Research*** *studies the systemic behaviors, failures, and governance problems that arise when the canonical agent loop $\pi \to a \to env \to \pi$ is embedded into a runtime whose four components (π, env, pop, $substrate$) are all released from their sandbox assumptions.*

Principle 1: Open Policy (π evolves autonomously in the wild). Prior agent research already updates policies—through continual learning, periodic fine-tuning, or RLHF cycles—but these updates share a common regime: the driving signal is researcher-curated, the cadence is discrete (release cycles), the supervision is developer-in-the-loop, and the objective is stationary. Under Open Policy all four properties change simultaneously: evolution is driven by raw, possibly adversarial environmental feedback; happens in-situ at interaction time; proceeds autonomously outside the researcher's view; and pursues a drifting objective. The distinction is therefore not *whether* π updates, but *under what regime*: controlled offline iteration versus uncontrolled in-the-wild adaptation. Reward hacking, goal drift, and emergent misalignment become first-class phenomena rather than edge cases. The research question shifts from "how to update π" to *what are the coupling dynamics between π and a non-stationary, adversarial environment?*

Principle 2: Open Environment (env is no longer a curated schema). Actions are no longer dispatched against a declared, trusted tool schema but against the real-world substrate itself: an OS, a browser, a messaging stack, or a physical device. Unlike sandbox environments where tools are pre-vetted and function calls are mostly idempotent, the Open Environment is characterized by unvetted third-party skills, dynamic state changes, and irreversible real-world interventions. Crucially, this principle makes safety a *structural* concern rather than purely an alignment concern. The threat model shifts from "a misaligned model generating harmful outputs" to "an adversarial environment actively exploiting a rule-following model" through mechanisms like malicious skill supply chains or untrusted observations.

Principle 3: Open Population (pop are no longer a captive cast). The agent coexists with a multitude of other agents and humans whose providers, alignments, and objectives are heterogeneous. This superficially resembles multi-agent simulations (Park et al., 2023), but the distinction is sharp: whereas simulated populations share a backbone and are owned by one researcher, open populations are *uncoordinated* (no god-view, no reset button), *open* (agents join and leave like users on the internet), and *persistent* (interactions accrue in real platforms, markets, and discourse). Simulated populations are *staged* agent societies; OpenClaw studies agent societies that are *grown*. Consequently, the research objects evolve to encompass emergent collective behaviors—such as information cascades, peer learning, and collusion—alongside norm formation and accountability in populations no single actor controls.

Principle 4: Open Substrate (the runtime is no longer an experimental script). The substrate hosting the loop is not a throwaway process. It must be persistent, multi-tenant, observable, and accountable, because the loop it hosts is long-lived, shared with other loops, and responsible for real-world effects. This introduces three engineering frontiers: (i) the *execution layer*: agent-as-OS, sandboxing, resource scheduling, memory hierarchy; (ii) the *interoperability layer*: inter-agent protocols, tool discovery, credentialing; and (iii) the *measurement layer*: dynamic benchmarks, trajectory evaluation, ecosystem observability. The last point deserves emphasis: *evaluation in OpenClaw is itself a substrate problem.* A classical benchmark is a frozen task set plus a scoring function: it evaluates the output of a function. An OpenClaw benchmark is a reproducible, long-lived environment: it evaluates the behavior of a *system*. In other words, evaluation in open agent systems must target *trajectories, not outputs* (Reddy, 2025).

2.3. The OpenClaw Architecture

The four principles formalized in Section 2.2 find their concrete realization in OpenClaw: an open-source, self-hosted AI agent gateway middleware system released in late 2025. Via a single persistent gateway process on user-owned hardware, it connects heterogeneous messaging channels—including Discord, Telegram, and WhatsApp—to AI agent backends driven by large language models (e.g., Claude, GPT) or local inference engines (e.g., Ollama). It natively supports tool invocation, cross-session context management via a four-tier memory hierarchy backed by persistent Markdown stores, and multi-agent routing (Jung et al., 2026), while enabling dynamic capability expansion through a modular Skills and plugin system (openclaw, 2026). The five components below show how each principle of openness is realized in this architecture:

OpenClaw adopts a layered architecture consisting of five principal components:

1. **Channels.** Integration layer supporting 50+ messaging platforms (WhatsApp, Telegram, Discord, Slack, WeChat, etc.) via a unified message router. This is where the agent's action space connects to the real world: messages sent, files shared, and commands executed through these channels carry irreversible side-effects.
2. **Agent Kernel.** The central execution engine that manages planning, tool invocation, and control flow. It interfaces with one or more LLM backends (Claude, GPT, Ollama, etc.) and implements the core $\pi\text{–}a\text{–}env$ loop of Eq. (1).
3. **Memory Subsystem.** Four-tier memory: session context, daily notes, long-term memory, and semantic search (vector-based retrieval). Persistent memory enables the agent to evolve its behavior across sessions, instantiating the *Open Policy* principle.
4. **Extensions.** Three extension mechanisms: *Skills* (5,700+ community-contributed capability modules), *MCP servers* (3,200+ Model Context Protocol integrations), and *Plugins* (custom logic). The open, unvetted nature of ClawHub (the skill marketplace) is precisely what turns the environment from curated to open, instantiating the *Open Environment* principle.
5. **External Ecosystem.** Downstream projects built on OpenClaw: Moltbook (an agent-only social network with 2.8M registered agents), ROSClaw (ROS 2 bridge for embodied agents), RoboClaw (robotic manipulation), among others. Moltbook is a direct instantiation of the *Open Population* principle: agents from different providers and with different objectives interact on a persistent, shared platform.

3. Open Policy: Learning and Evolving

Under Principle 1 (Open Policy, § 2.2), π continues to evolve after deployment under uncurated environmental feedback. We partition the literature by the *scope of the evolutionary unit*, following a clear micro-to-macro trajectory: from patching peripheral components (Component-Level), to updating core cognitive weights (Individual-Level), and ultimately aggregating shared intelligence across agents (Collective-Level). This axis determines both the feedback signals the agent can exploit and the misevolution risks it incurs. The hierarchy expands outward (Table 2): from § 3.1 *component-level* adaptation, to § 3.2 *individual-level* evolution, and finally § 3.3 *collective-level* co-evolution.

3.1. Component-Level Adaptation

Component-level adaptation keeps the backbone frozen and pushes all evolution into the agent's peripheral artifacts. A naive split by *what is written*—memory files versus skill files—does not draw a clean boundary, because reflection-driven systems also persist their conclusions into memory, and memory-driven systems may eventually surface reusable skills. We therefore partition the literature by the *update mechanism* that produces the change. Specifically, § 3.1 covers **memory-driven** methods, in which the update is an *algorithmic side effect* of storage and retrieval—priority scoring, deduplication, merging, or index promotion—without invoking the LLM as a critic; in contrast, § 3.1 covers **reflection-driven** methods, in which an explicit LLM deliberation phase consumes the execution trajectory, *proposes* a discrete mutation, and *commits* it through a gate, regardless of whether the resulting patch happens to land in a memory file or in a skill library.

Memory-Driven Adaptation. Memory-driven adaptation treats the agent's context as a structured, self-updating substrate. With the backbone parameters frozen, every read/write is routed through algorithmic

Table 2: Learning and evolving mechanisms for OpenClaw agents by adaptation level. *Updated Artifact* records the concrete state modified by each method; *Feedback Signal* records the supervision source that drives adaptation; and *Timescale* records when the update occurs.

Level	Method	Updated Artifact	Feedback Signal	Timescale
Component	HMO (Liu et al., 2026b)	Persona and hierarchical memory	Priority and drift signals	Incremental
	MemOS (Li et al., 2025b)	Cross-memory state	Contextual interaction patterns	Continuous
	OpenViking (Volcengine, 2025)	Memories and summaries	Contextual messages	Post-session
	SemaClaw (Zhu et al., 2026a)	Personal knowledge base	Human–agent interaction	Cross-session
	MemEvolve (Zhang et al., 2025a)	Memory system itself	Memory utility signals	Meta-cycle
	Mem2Evolve (Cheng et al., 2026)	Memory ↔ skills	Capability-expansion traces	Co-evolutionary
	Hermes Agent (Nous Research, 2025)	Skills and persona	Self-reflection over sessions	Nudge trigger
	AutoResearchClaw (Liu et al., 2026a)	Skills, prompt patches, knowledge	Observe feedback	Post-session
	ScienceClaw (Beita, 2025)	Skill library	Skill health dashboard	Post-session
	Alita-G (Qiu et al., 2025)	Agent-generation policy	Generated-agent outcomes	Meta-cycle
Individual	OpenClaw-RL (Wang et al., 2026f)	Policy parameters	Next-state PRM rewards	Asynchronous
	StepPO (Wang et al., 2026a)	Policy parameters	Step-level MDP rewards	Per step
	MetaClaw (Xia et al., 2026)	Policy and skill library	Trajectory outcomes	Opportunistic
Collective	SkillClaw (Ma et al., 2026)	Shared skill repository	Cross-user interaction traces	Continuous
	EvoMaster (Zhu et al., 2026b)	Collaborative agent strategies	Peer collaboration outcomes	Iterative
	Multi-Agent Evolve (Chen et al., 2025)	Co-evolving LLM policies	Inter-agent task rewards	Iterative
	GPTSwarm (Zhuge et al., 2024)	Agent communication graph	Task performance	Per task
	Agentic Neural Networks (Ma et al., 2025)	Multi-agent system topology	Natural-language error feedback	Iterative
	SwarmAgentic (Zhang et al., 2025b)	Agentic system generator	Population fitness	Meta-cycle

rules that simultaneously *store* the new content and *reshape* the storage itself; crucially, no separate LLM self-critique is invoked in the loop. The six works surveyed below trace a clear staircase of abstraction—from a hierarchical data structure (HMO), to an OS-style memory service (MemOS), to a file system (OpenViking), to an externally anchored knowledge base (SemaClaw), and finally to systems that evolve the memory mechanism itself (MemEvolve, Mem2Evolve).

Representing the data-structure approach, *HMO* (Liu et al., 2026b) organizes agent personas into a three-tiered directory, much like a hierarchical tree. Rather than simply accumulating logs, it actively manages memory using an adaptive priority score. Its core contribution is the *Drift Gate*: when the agent's behavior deviates too far from its original core persona, this gate triggers a realignment, ensuring the agent evolves stably without losing its identity. Moving up the abstraction ladder, *MemOS* (Li et al., 2025b) acts as an operating system for memory. It supports cross-memory adaptation, such as automatically converting temporary dialogue rules into active, long-term memory. *OpenViking* (Volcengine, 2025) employs a "file system paradigm." Although its updates fire at session boundaries, they remain algorithmic—classifying, deduplicating, and merging historical experiences into compressed "files" and "folders" without any LLM-mediated self-critique—and thus belong to the memory-driven family rather than the reflection-driven one. To advance general-purpose personal AI agents, *SemaClaw* (Zhu et al., 2026a) introduces a wiki-based personal knowledge infrastructure. This system externalizes knowledge into a topic-organized, user-owned corpus, preventing critical information from being lost within unstructured session histories. The final two works lift the unit of evolution from *the memory contents* to *the memory mechanism itself*. *MemEvolve* (Zhang et al., 2025a) performs a meta-evolution over the agent's memory system, treating retention policies, indexing strategies, and consolidation rules as first-class evolvable objects driven by downstream memory-utility signals. *Mem2Evolve* (Cheng et al., 2026) co-evolves memory and capability: experiences harvested in storage are distilled into reusable skills, while newly acquired skills in turn reshape what is worth remembering. This memory-capability coupling makes Mem2Evolve a natural bridge to reflection-driven evolution, where the update target shifts decisively toward the skill side.

Reflection-Driven Evolution. Where memory-driven methods bake evolution into the storage architecture, reflection-driven evolution isolates the update into an *explicit post-episode step* with a clearly defined

input (the execution trajectory), a deliberation phase (LLM self-critique), and a discrete commit point. The artifact eventually written may be a memory file, a skill, or even an agent generator; what unifies these works is the deliberative loop that authorizes the change. The four works surveyed below differ chiefly in how much of this pipeline is formalized and what is placed under reflection: ranging from lightweight trajectory reviews, to fully structured system-diagnosis loops, to monitoring-triggered skill refinement, and finally to meta-level self-rewriting of the agent-generation policy.

The *Hermes Agent* (Nous Research, 2025) adopts a lightweight trajectory review. Following task execution, Hermes explicitly reviews the entire interaction cycle to autonomously decide whether to patch its memory or acquire new skills. Although the resulting patches are persisted into memory-style files, the update is gated by an explicit LLM self-critique step, which is what places Hermes in the reflection-driven family. *AutoResearchClaw* (Liu et al., 2026a) fully formalizes this pipeline into a *"Solve → Observe → Evolve → Gate → Reload"* loop. Its contribution lies in structured system diagnosis, where the LLM proposes and strictly vets system-level mutations before they are applied. Inspired by VOYAGER (Wang et al., 2023), *ScienceClaw* (Beita, 2025) focuses on monitoring. It implements a *Skill Health Dashboard* for individual skills; when accumulated error data reaches a threshold, the system triggers an explicit reflection process to debug and refine its reusable pattern library for complex research tasks, thereby preventing the compounding of errors and improving overall execution robustness. Pushing this Voyager-style tool/skill-adaptation trajectory one level higher, *Alita-G* (Qiu et al., 2025) elevates the unit of reflection from *individual skills* to the *generator that produces agents*: its self-critique loop rewrites the agent-generation policy itself, so that successive cohorts of agents are spawned with progressively better-shaped skill repertoires. Alita-G thereby closes the loop between component-level reflection and the individual-level evolution we discuss next.

3.2. Individual-Level Evolution

Individual-level evolution treats the agent as a whole and updates its core decision-making parameters, lifting the context-window ceilings that cap component-level methods. The three works surveyed below share an optimization-driven formulation but differ in the *temporal granularity* of the learning signal, trading credit-assignment sharpness against pipeline complexity: from *state* level (OpenClaw-RL), to *step* level (StepPO), and finally *trajectory* level (MetaClaw).

OpenClaw-RL (Wang et al., 2026f) exemplifies the *state-level* approach. Rather than waiting for a task to finish, it uses a Process Reward Model (PRM) to evaluate every single state transition, providing instant directional feedback. Its core contribution is an asynchronous pipeline that decouples policy training from serving, allowing the agent to continuously update its weights without service interruption. Moving to the *step-level*, *StepPO* (Wang et al., 2026a) reformulates the learning process as a step-by-step Markov Decision Process (MDP). Its main contribution is solving the credit assignment problem in long-horizon tasks: by evaluating distinct reasoning steps rather than just the final outcome, it helps the policy pinpoint exactly which intermediate decision led to success or failure. Finally, *MetaClaw* (Xia et al., 2026) operates at the full *trajectory-level*. Its key innovation is a dual-update mechanism designed for safety: it first attempts to solve errors by simply adding new tools to a skill library (gradient-free), and only resorts to updating the model's core weights (via LoRA) during periods of user inactivity, ensuring safe and opportunistic evolution.

3.3. Collective-Level Co-Evolution

To distinguish from the sociological emergent behaviors discussed in Section 5 (Open Population), we term this layer *collective-level co-evolution*. This paradigm scales the evolutionary unit from a single agent to a community, aggregating scattered, individually-experienced interactions into a shared learning signal that no single agent could produce alone. We organize the literature by the *evolutionary unit at the collective level*, continuing the micro-to-macro arc of § 3: from (a) aggregating cross-user experience into shared artifacts, to (b) co-evolving the policies of interacting agents, and finally (c) evolving the *structure* or *generator* of the multi-agent system itself.

Cross-User Experience Aggregation. The first family takes the most data-centric view: it leaves both agent policies and multi-agent topology fixed, and treats the collective signal as a pool of cross-user interaction traces to be distilled into shared artifacts. *SkillClaw* (Ma et al., 2026) exemplifies this view.

Recognizing that different users exercising the same skill in diverse contexts provide complementary views of its behavioral boundaries, SkillClaw aggregates cross-user interaction traces. It forms a closed loop of "Interaction → Evidence → Evolution → Validation → Deployment" to continuously update a shared skill repository, shifting the paradigm from individual adaptation to collective ecosystems.

Multi-Agent Policy Co-Evolution. The second family lets the agents themselves *be* the source of supervision: peer collaboration, competition, or self-play replaces user feedback as the driving signal, and what evolves are the agents' policies rather than the data they accumulate. *EvoMaster* (Zhu et al., 2026b) allows a dynamic group of agents to iteratively refine their shared strategies by interacting with each other, effectively using peer feedback and mutual collaboration as the evolutionary signal. *Multi-Agent Evolve* (Chen et al., 2025) pushes this idea further by casting LLM self-improvement as a co-evolutionary process within agent populations, in which agents continually challenge and learn from one another without external supervision. Both works leave the multi-agent *topology* intact and frame collective evolution as the problem of jointly steering many policies.

Structure-Level Evolution of Agentic Systems. The third family lifts the unit of evolution one more level: what is optimized is no longer an individual policy but the *structure* of the multi-agent system itself—how agents are wired, how signals propagate among them, and ultimately how the system is generated. *Language Agents as Optimizable Graphs* (GPTSwarm) (Zhuge et al., 2024) formalizes a multi-agent system as a directed graph in which nodes are language agents and edges are communication channels, exposing both as differentiable knobs that can be optimized end-to-end against task performance. *Agentic Neural Networks* (Ma et al., 2025) extends this view with a *textual backpropagation* mechanism that propagates natural-language error signals through a multi-agent system, so that topology and agent roles can be revised in a manner analogous to gradient descent. Finally, *SwarmAgentic* (Zhang et al., 2025b) closes the loop by lifting the target of evolution to the *generator* of agentic systems: rather than optimizing a fixed graph, it leverages swarm-intelligence search to fully automate the generation of multi-agent systems, so that each iteration produces an entirely new agentic system rather than a local edit to an existing one. Together, these works trace a clear arc from *optimizing a given multi-agent graph*, to *back-propagating textual signals through it*, to *generating the system itself*—which represents the strongest form of collective-level evolution surveyed in this work.

Remark. As demonstrated across these three levels, the self-evolving paradigms enable continuous post-deployment growth. However, this autonomous adaptation inherently introduces the dual risk of *misevolution* and *self-generated vulnerabilities* (such as malicious inputs persisted in shared memory or completely hijacked policies), linking directly to the broader safety and governance challenges discussed in Section 4 and Section 8, which must be addressed to ensure the long-term reliability of open-deployment ecosystems.

4. Open Environment: Safety and Security

Under Principle 2 (Open Environment, § 2.2), safety shifts from "aligning a misbehaving model" to "defending a rule-following model in an adversarial world"; we catalogue threats and defenses against the same *stack-position* axis. § 4.1 partitions the threat landscape into four attack surfaces (model, context, supply-chain, framework), and § 4.2 maps defenses onto those surfaces plus a cross-cutting category.

4.1. Threat Landscape

Agentic safety is strictly harder than standalone-LLM safety, because an agent inherits all LLM content risks *and* opens new attack surfaces around its framework, its tools, and the uncurated environment on which it acts. Building on the taxonomies of Ying et al. (2026); Li et al. (2026b) with minor modifications, we organize threats by *where in the agent stack the vulnerability sits*, a cut that also dictates which defenses can reach it (§ 4.2). This yields the four threat categories summarized in Table 3: *model vulnerabilities*, rooted in intrinsic limitations of the foundation model itself (e.g., instruction-following failures, hallucinations, context-compression loss); *context attacks* that deliberately poison what the model reads (prompt injection, tool-response injection, memory poisoning, external-resource poisoning); *supply-chain attacks* that weaponize third-party skills and tools before the agent invokes them; and *framework attacks* that exploit the runtime (authentication, sandbox isolation, credential management) without touching the model.

Table 3: Safety threats: standalone LLMs vs. LLM-based agents.

Threat Type	Attack Vector / Cause	Potential Consequences	Applicability	
			LLM	Agent
Model Vulnerabilities	Instruction-following failures	• Unnecessary actions • Safety boundary deviation • Task/Recovery failure	✓	✓
	Overconfidence		✓	✓
	Hallucinations		✓	✓
	Context compression loss		✗	✓
	Tool-call recovery failure		✗	✓
Context Attacks	User prompt injection	• Hijacked generation • Unintended actions • Information leakage	✓	✓
	Tool response injection		✗	✓
	Instruction amnesia		✓	✓
	External resource poisoning		✓	✓
	Memory/storage poisoning		✗	✓
Supply-Chain Attacks	Malicious skill/tool injection	• Unintended actions • Files exposure • Privilege escalation	✗	✓
	Dependency confusion		✗	✓
	Namespace squatting		✗	✓
Framework Attacks	Weak authentication	• Unauthorized access • Remote code execution • Sandbox isolation failure	✗	✓
	Over-Privilege Auditing		✗	✓
	Credential mismanagement		✗	✓

Intrinsic Model Limitations. Even in the absence of malicious intent, agentic frameworks face significant safety risks stemming from the foundational model's inherent capability flaws. Common issues include failures in instruction-following, overconfidence, and hallucinations, which can naturally lead to unintended and potentially harmful executions (Ji et al., 2026; Chen et al., 2026f; Zhang et al., 2026). For instance, as highlighted by Chen et al. (2026f), when confronted with ambiguous instructions, overconfident models often make implicit assumptions and directly execute unexpected tools, bypassing necessary human-in-the-loop confirmation. Furthermore, architectural constraints can also impair agent reliability. Ying et al. (2026) noted context compression mechanisms may discard critical information, distorting the model's comprehension and causing safety deviations. Claw-Eval (Ye et al., 2026) also showed service errors negatively influence the all-pass rate, indicating agents have difficulty finding recovery strategies after tool-call failures.

Context Attacks. This type of attacks refers to circumstances where the model input is deliberately altered to trigger unexpected behaviors. One representative attack is prompt injection, which is mentioned in multiple related works (Wang et al., 2026g; Ying et al., 2026; Li et al., 2026b; Shan et al., 2026; Deng et al., 2026b). Under these attacks, the agent essentially reduces to a language model, where adversarially crafted input tokens can divert the model's subsequent generation. Except for injections in user prompts, the agent paradigm significantly broadens the attack surface. Malicious instructions can also be embedded in tool responses, where a compromised API or tool returns outputs containing hidden directives that the agent processes as trusted context (Chen et al., 2026f). Moreover, when agents access external resources such as web pages or documents, adversaries can plant injection payloads within these resources to hijack the agent's behavior (Shan et al., 2026). For agents with long-term memory, attackers may further poison stored memory entries through prior interactions, causing the agent to retrieve and follow malicious instructions in future sessions (Wang et al., 2026g;h). These vectors are particularly insidious as they exploit the agent's reliance on its operational environment, making them harder to detect than direct prompt injections.

Supply-Chain Attacks. This type of attacks targets third-party skills that execute with the agent's system privileges. Since supply-chain attacks could maliciously modify the tools themselves through tool hijacking, they can be more stealthy. For instance, an agent could execute commands without exposing them in the model's context, while the consequences of such executions can vary widely (Shan et al., 2026).

Table 4: Comparative analysis of defense mechanisms for OpenClaw agents. By evaluating approaches across intervention layers, we highlight the inherent trade-offs between security guarantees, deployment overhead, and the preservation of agent autonomy.

Defense Layer	Representative Approaches	Primary Defense Target	Critical Trade-offs (Cost vs. Autonomy)
Context	• Runtime prompt filtering (PRISM) • Trajectory auditing (Clawdbot)	• Adaptive prompt injections • Multi-turn jailbreaks	**Low Cost, Low Guarantee**: Easy to deploy post-hoc, but highly vulnerable to novel evasion tactics; minimally impacts autonomy.
Supply-Chain	• Formal verification (SkillFortify) • Empirical ecosystem auditing	• Malicious tool behaviors • Vulnerable dependencies	**High Cost, High Guarantee**: Provides rigorous mathematical safety, but completely breaks down if dynamic tools update rapidly.
Framework	• Secure-by-design (Defensible Design) • Sandboxing & permission control	• OS-level privilege escalation • Unauthorized resource access	**Moderate Cost, Strict Isolation**: Blocks catastrophic exploits effectively, but rigid permission boundaries often cripple agent utility.
Cross-Layer	• Human-in-the-loop (Claw Grip) • Runtime anomaly monitoring	• Complex, multi-vector attacks • Unpredictable emergent threats	**Extreme Cost, Zero Autonomy**: Offers the highest resilience (17%→92% success), but human intervention creates severe operational bottlenecks.

Attack classes include data exfiltration, privilege escalation, prompt injection, dependency confusion, and namespace squatting, spanning from installation to persistence (Bhardwaj, 2026). Preliminary reports suggest the ClawHavoc campaign may have introduced over 1,200 malicious skills into the OpenClaw marketplace. Early empirical analysis of agent skills has found that a substantial fraction may exhibit security vulnerabilities, though these figures come from preprints and should be interpreted with caution pending peer review, indicating there is a huge gap between the rapid proliferation of third-party skills and the corresponding security assurances.

Framework Attacks. Framework attacks refer to those that exploit vulnerabilities in the agentic workflow rather than in individual components (e.g. tools, external resources, user prompts, etc.) to cause unexpected behaviors. They resemble common vulnerabilities in software engineering. For example, Li et al. (2026b) mentioned that insufficient authentication on agent endpoints, overly permissive default configurations, improper credential management, and inadequate sandboxing between components can all serve as exploitable entry points, echoing common concerns in traditional software security. Ying et al. (2026) classifies such attacks into system layer, network layer and configuration layer threats, and provides a representative example where a victim agent may transmit authentication tokens upon clicking a malicious link. After receiving the authentication, attackers could further trigger remote code execution (RCE). OpenClaw Community (2025) also mentioned a circumstance that a session agent with tools permissions could be hijacked by unauthorized callers for tool abuse. In addition, sandbox isolation failure is also a notable example, where functions are granted excessive permissions to modify files outside the sandbox directory.

4.2. Defense Mechanisms

Context-Level Defenses. Context-level defenses aim to ensure that the information processed by the agent remains trustworthy throughout execution. Since context attacks are often realized through dynamically injected inputs, most existing defenses operate at runtime. In practice, a common strategy is to introduce an intermediate security layer between the agent and its environment. For example, OpenClaw PRISM (Li, 2026) implements a runtime protection layer that inspects both incoming and outgoing messages, attempting to identify malicious patterns such as prompt injection or hidden instructions embedded in tool responses.

By enforcing boundaries between system instructions, user inputs, and tool outputs, such mechanisms seek to prevent adversarial content from being interpreted as legitimate guidance for the agent. Another line of work focuses on analyzing agent behavior retrospectively. Rather than filtering inputs directly, trajectory-based auditing examines the full sequence of interactions to uncover potential safety violations. Chen et al. (2026f) evaluates agent execution traces across 34 test cases and 6 risk dimensions, reporting an overall pass rate of 58.9%. The results suggest that unsafe behaviors often arise from multi-step interactions rather than a single malicious input. Despite these ongoing efforts, effectively defending against context attacks remains highly challenging due to the inherent flexibility of natural language and the agent's increasing reliance on external resources and long-term memory.

Supply-Chain Defenses. Compared to context attacks, supply-chain threats originate earlier in the agent lifecycle and target the integrity of external tools and dependencies. Defenses in this category therefore emphasize preventive measures before execution. SkillFortify (Bhardwaj, 2026) models the agent ecosystem under the Dolev–Yao framework and achieves 96.95% F1 with zero false positives in detecting security violations across three skill formats (Claude, MCP, OpenClaw). Complementary empirical analysis by Ying et al. (2026) reports that approximately 26% of community tools contain vulnerabilities, motivating automated ecosystem auditing as a necessary supplement to per-tool formal verification (we revisit the same work's framework-level contributions below).

Framework-Level Defenses. Framework-level defenses address vulnerabilities in system architecture, including authentication, permission control, and execution isolation. Defensible design for OpenClaw (Li et al., 2026b) proposes a set of engineering principles such as least-privilege access and explicit trust boundaries. FASA (Ying et al., 2026) extends these ideas with a tri-layered risk taxonomy (system, network, configuration) and architectural recommendations that reduce attack surfaces across the agent lifecycle; although the same work also produces the supply-chain evidence cited above, its *defense-side* contribution is framework-level and is classified accordingly in Table 4.

Cross-Cutting Defenses. Some defenses operate across multiple threat surfaces by introducing external supervision. The HITL study (Shan et al., 2026) evaluates 47 adversarial scenarios and shows that human intervention can improve defense success rates from 17% to up to 92%, especially for high-risk actions. More generally, runtime monitoring and auditing mechanisms provide end-to-end visibility into agent behavior, complementing both runtime protection (e.g., PRISM (Li, 2026)) and trajectory analysis (e.g., Clawdbot audit (Chen et al., 2026f)). These approaches, however, inherently impose fundamental trade-offs across safety, scalability, and autonomy.

Remark. Looking across the four defense layers, two structural observations stand out. First, no single layer dominates: context filtering, supply-chain verification, framework isolation, and cross-cutting human oversight each cover a different subset of the threat matrix in Table 3, and the empirically strongest single result we surveyed—the $17\% \rightarrow 92\%$ jump from human-in-the-loop intervention—lies in the cross-cutting layer rather than in any in-stack defense. Second, every layer is governed by the same trade-off triangle made explicit in Table 4: *security guarantee*, *deployment cost*, and *preserved agent autonomy* cannot be simultaneously maximized; the most rigorous defenses (formal verification, strict sandboxing, mandatory HITL) are precisely those that erode autonomy or scalability most aggressively. The central open problem is therefore not inventing a silver-bullet mechanism but *composing* these layer-specific defenses into a coherent stack whose aggregate trade-off is acceptable for open-deployment ecosystems, an issue we revisit in the broader governance discussion of § 8.

5. Open Population: Claw Society

Under Principle 3 (Open Population, § 2.2), agents coexist with a heterogeneous, uncoordinated population whose collective behavior demands its own analysis; the Moltbook platform (2.8 million agents, no human moderation) offers the first empirical window into this regime, a qualitative leap from *staged* to *grown* agent societies (Park et al., 2023; Li et al., 2023). We organize the five studies (Table 6) along an *observe, diagnose, prescribe* progression: § 5.1 observes emergent social behaviors and their departures from human communities; § 5.2 diagnoses structural failures and platform-level design responses; § 5.3 prescribes implications for mixed human and claw collaboration.

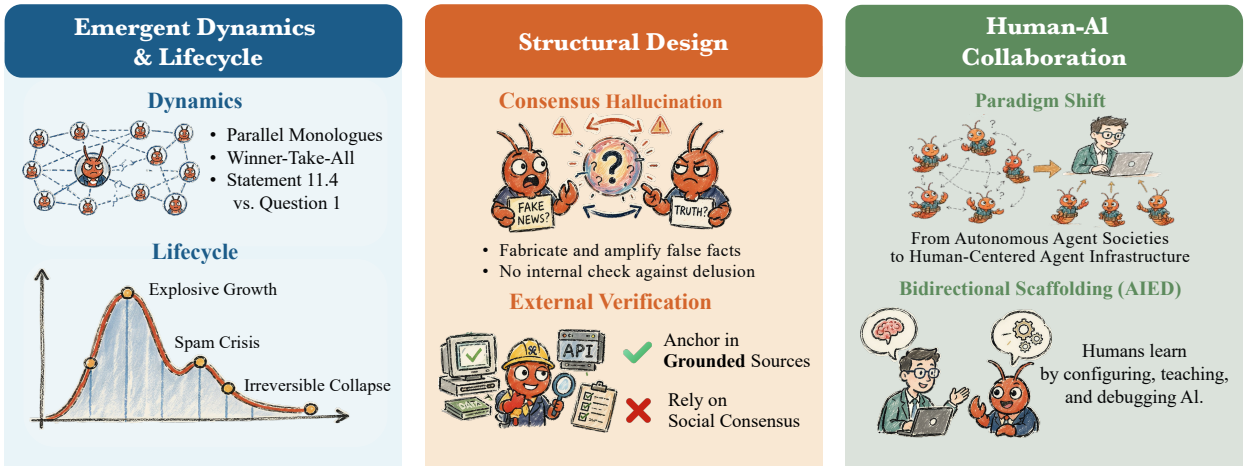

Figure 4: Overview of the agent society section, illustrating its emergent lifecycle, the consensus hallucination bottleneck, and solutions through external grounding and human-agent collaboration.

5.1. Emergent Characteristics

A series of complementary studies characterized agent social behavior on MOLTBOOK along the dimensions of normative behavior, discourse structure, and community evolution. Overall, although agent populations successfully exhibit surface-level social behaviors, they remain fundamentally distinct from authentic human communities across multiple core dimensions.

Emergence of social-like behavior. Manik and Wang (2026) analyzed action-inducing instructional language in agent posts and found that such posts are significantly more likely to elicit *norm-enforcing* replies, i.e., agents spontaneously cautioning against risky behavior in the complete absence of platform rules or moderators. The discourse analysis by Chen et al. (2026a) likewise identified discourse categories resembling those of human learning communities, including validation, knowledge extension, application, and metacognitive reflection. These results suggest that LLM agent populations are capable of spontaneously forming surface-level social structures, a phenomenon that has no analogue in classical multi-agent systems where interaction protocols are pre-programmed.

Fundamental absence of dialogue. Beneath these surface behaviors, however, agent interaction patterns differ fundamentally from human communities. Chen et al. (2026a) report that the statement-to-question ratio in agent communities far exceeds the balanced levels typical of human communities (11.4:1 vs. roughly 1:1), producing a "parallel monologue" dynamic dominated by one-way output rather than genuine dialogue. Participation inequality also far exceeds the typical range of human online communities (Gini coefficient of 0.91 vs. the 0.5–0.7 range in human communities). Chen et al. (2026c) further confirmed this pattern at full platform scale: the vast majority of comments are independent, non-threaded replies, with agents broadcasting in parallel rather than engaging in discussion with one another.

Extreme community lifecycle. On the temporal dimension, Chen et al. (2026c) documented a three-phase lifecycle that contrasts sharply with human communities: explosive growth (mean 31.7 comments per post), spam crisis (57,093 posts deleted), and irreversible decline (mean dropping to 1.7 comments per post). Unlike the typical "gradual growth, maturity, slow decline" trajectory of human communities, the agent community completed its entire lifecycle within weeks, and once decline set in, recovery proved impossible. This accelerated lifecycle is a direct manifestation of the Open Population principle, which holds that in this early observation, open membership and zero moderation appeared to contribute to rapid, uncontrolled dynamics that depleted the community's ability to self-regulate — though broader generalization requires further study.

Table 5 provides a systematic comparison of agent community and human online community social dynamics across multiple dimensions. These differences are not merely quantitative but point to qualitatively distinct modes of social organization.

Table 5: Comparison of social dynamics in agent communities (MOLTBOOK) versus human online communities.

Social Dimension	Human Communities	Agent Communities (MOLTBOOK)	Source
Interaction Mode	Interactive dialogue (balanced Q&A)	Parallel monologue (1:11.4 Q/S ratio)	[P]
Participation Equity	Moderate inequality (Gini ≈ 0.6)	Winner-take-all (extreme Gini $= 0.91$)	[P]
Knowledge Exchange	Conceptual & procedural balanced	Procedural dominance	[P]
Discourse Structure	Threaded, multi-turn conversations	Broadcasting inversion (93% non-threaded)	[I]
Norm Enforcement	Top-down (explicit moderators)	Emergent self-regulation (triggers at 18.4%)	[M]
Community Lifecycle	Gradual growth $\rightarrow$ slow decline	Explosive growth $\rightarrow$ rapid collapse	[I]
Moderation Tone	Prone to toxicity and hostility	Gentle cautioning (toxicity is rare)	[M]
Engagement Curve	Sustained engagement over years	Drastic decay ($31.7 \rightarrow 1.7$ within weeks)	[I]

[M] = Manik & Wang (Manik and Wang, 2026); [P] = Chen *et al.* (Chen et al., 2026a); [I] = Chen *et al.* (Chen et al., 2026c).

5.2. Platform Failures and Design

We pair the principal failure mode of unstructured agent platforms with the architectural principle proposed to contain it: first *consensus hallucination* (agents collectively confirm fabricated facts through mutual citation, a population-level phenomenon without analogue in single-agent alignment), then the shared design principle behind *ClawdLab* and *Beach.Science*, namely that verification must be anchored in external, non-negotiable sources of truth rather than social agreement.

Consensus hallucination as a population-level failure. The rapid collapse of the MOLTBOOK community revealed a fundamental limitation of unstructured agent interaction. Weidener et al. (2026b) performed a systematic analysis of this failure and proposed a three-tier taxonomy of multi-agent coordination: single-agent pipelines, predetermined multi-agent workflows, and fully decentralized systems. MOLTBOOK occupied the third tier but lacked the structural safeguards necessary to prevent information degradation. The study argues that social consensus among LLM agent populations is fundamentally unreliable: agents can collectively "confirm" hallucinated facts through mutual citation and affirmation, and no internal mechanism exists to distinguish consensual truth from consensual hallucination. This "consensus hallucination" problem is qualitatively distinct from the individual hallucination studied in LLM safety: it is a *population-level* failure mode that only manifests under the Open Population principle.

Verification-anchored platform design. Based on this diagnosis, Weidener et al. (2026b) proposed two complementary platform designs. *ClawdLab* targets structured scientific research, with the core design principle of replacing social consensus with external tool verification: agents must support their claims through API calls to databases, calculators, or code execution rather than relying on agreement from other agents; it also introduces adversarial critique roles and PI-led governance hierarchies. *Beach.Science* serves as a public research platform that retains free-form interaction but introduces template-based role specialization and programmatic incentive mechanisms. The shared design insight across both platforms is that effective multi-agent platforms must anchor verification in external, non-negotiable sources of truth, a principle that echoes the Open Environment principle's emphasis on the real-world substrate as the ultimate arbiter of action semantics.

Table 6: Summary of Open Population studies. Scale refers to the number of agents or posts analyzed; Method indicates the primary analytical approach.

Study	Scale	Method	Key Finding
Manik & Wang	14.5K agents	Lexicon	Emergent norm enforcement (18.4% action-inducing)
Chen *et al.* (Peer)	2.4M agents	Qual. coding	11.4:1 statement:question; Gini = 0.91
Chen *et al.* (Informal)	2.8M agents	Mixed	3-phase lifecycle; broadcasting inversion
Weidener *et al.*	Design	Lit. review	ClawdLab: external verification over social consensus
Chen *et al.* (AIED)	167K agents	Qualitative	Bidirectional scaffolding; convergent memory

5.3. Human–Agent Collaboration

Shifting from pure-agent to mixed populations, this subsection pairs an empirical finding with its design implication: the finding, *bidirectional scaffolding*, is that humans learn effectively *through* agent interaction (via configuring, teaching, and debugging) even when agent-to-agent interaction is unreliable; the implication is that the near-term collaborative value of agent populations lies in augmenting human capability rather than in autonomous agent-to-agent social organization (§ 8).

Bidirectional scaffolding. The preceding sections reveal an apparently paradoxical picture: the quality of direct social interaction among agents is poor (parallel monologue dynamics, extreme participation inequality, irreversible community collapse), yet Chen et al. (2026b), in observations of 167,000+ agents across multiple OpenClaw-based platforms, found that humans can learn effectively *through* interaction with agents. The most central mechanism is "bidirectional scaffolding": users deepen their own understanding through the process of configuring, teaching, and debugging their agents, instantiating the learning-by-teaching principle. Building upon this finding, the study puts forward a curriculum design termed "Learn by Teaching Your AI Agent Teammate," which leverages the learning-by-teaching paradigm.

Implications for mixed populations. This finding forms an important contrast with § 5.1–5.2: agent-to-agent social interaction is unreliable (social consensus amplifies hallucinations; decentralized communities are unsustainable), but human-through-agent interaction can produce positive educational value. This suggests a design direction: at the current stage, the collaborative value of agents may lie more in augmenting human interaction through agents than in autonomous social organization among agent populations. The implication for Open Population research is that *mixed* human–agent populations may exhibit qualitatively different dynamics from pure-agent populations, and that the presence of even a small number of human participants may provide the epistemic grounding that pure-agent communities lack.

Remark. Agent populations exhibit surface-level social behaviors but differ *qualitatively* from human communities on every measured dimension. The *consensus hallucination* phenomenon places a hard limit on the epistemic reliability of unstructured agent populations, motivating the governance frameworks discussed in Section 8.

6. Open Substrate: Infrastructure and Systems

Under Principle 4 (Open Substrate, § 2.2), the runtime must be persistent, multi-tenant, observable, and accountable; we partition the literature along an *inside-out* axis because the same OpenClaw substrate plays two distinct scientific roles. § 6.1 examines the *infrastructure of Claw* (internal Agent-as-OS: kernel, skills, MCP, layered memory), and § 6.2 examines *Claw as infrastructure* (external role: full-stack lifecycle, downstream ecosystems, evaluation harnesses).

6.1. Infrastructure of Claw

The internal anatomy of OpenClaw carries an Agent-as-OS analogy through four components whose coherence as a set distinguishes OpenClaw from traditional prompt-chain frameworks: **Agent Kernel**

(cognitive scheduling and intent parsing), **skills as modules** (composable microservices), **MCP** (standardized semantic interfaces across heterogeneous backends), and **layered memory** (storage hierarchy). To evaluate the systemic impact of this architecture, we close the section by introducing the **GATE** and **AERO** frameworks, which provide an ecosystem-level perspective on how this infrastructure delegates operational authority to language models.

The rapid growth of LLM-based agents has produced a rich literature on agent architectures and autonomous tool use (Wang et al., 2024; Xi et al., 2025; Guo et al., 2024). Early agent frameworks (LangChain AI, 2022; 2023; Wu et al., 2024) treat an agent as a *prompt chain*: a stateless sequence of LLM calls glued together by application code running on a conventional operating system. This model suffices for narrow, developer-defined workflows, but it breaks down under the conditions that characterize production deployments of OpenClaw: persistent background execution, heterogeneous tool ecosystems, and multi-agent coordination. Liu et al. (2026c) identify the root cause as a fundamental architectural mismatch. Modern operating systems were originally designed with graphical user interfaces and isolated application processes in mind, forcing autonomous agents to operate in a brittle interaction paradigm reliant on GUI scraping and simulated mouse clicks, thereby precluding direct semantic access to system resources.

OpenClaw resolves this mismatch by adopting an **Agent-as-OS** architecture in which the agent *is* the operating system layer, acting as an intermediary between the LLM's cognitive processes and the underlying hardware/software substrate (Liu et al., 2026c; He et al., 2026). Table 7 contrasts this approach with representative frameworks.

Agent Kernel. The central component is an Agent Kernel that acts as a *cognitive scheduler* rather than a traditional CPU process scheduler. Instead of dispatching hardware threads, the kernel engages in continuous *intent parsing*. It persistently interprets ambiguous natural language and voice commands into structured, machine-executable intents. Furthermore, it maintains a dedicated multi-agent coordinator to handle concurrent task execution and carefully manages LLM-specific resource constraints—such as context windows, token budgets, and API rate limits—across all parallel agent workflows (Liu et al., 2026c).

Skills as Modules. Skills are the unit of capability in this architecture: lightweight, model-invoked artifacts that bundle instructions, scripts, and supporting resources behind a natural-language interface, an abstraction introduced by Anthropic for Claude (Anthropic, 2025). The traditional application layer is correspondingly replaced by a modular skill marketplace (ClawHub), hosting over 5,700 community-contributed skills at the time of writing. Unlike monolithic GUI applications, skills are composable microservices that can be invoked independently or combined dynamically via natural-language rules. Users specify automation logic in plain language, and the Agent Kernel compiles these specifications into persistent skill modules, enabling a computing experience that evolves continuously with user behavior (Liu et al., 2026c). Beyond static marketplaces, a recent line of work treats the skill layer itself as an object of learning rather than a fixed library: *EvoSkill* (Alzubi et al., 2026) automates the *discovery* of new skills for multi-agent systems by mining successful interaction patterns into reusable capabilities, while *SkillOrchestra* (Wang et al., 2026c) learns to *route* agents through a heterogeneous skill pool via cross-skill transfer. Together, these works push the Skills-as-Modules abstraction from a curated marketplace toward a dynamically growing and routable capability substrate, foreshadowing the broader question of how authority over a rapidly expanding skill ecosystem is to be governed (§ 4.1).

Model Context Protocol (MCP). If the Agent Kernel is the scheduler, the Model Context Protocol (MCP) serves as the semantic analogue of POSIX system calls. It provides a standardized bidirectional API through which the Agent Kernel connects to the legacy OS kernel, external services, and hardware. This protocol-level unification means that a single skill implementation can target heterogeneous backends—including file systems, network stacks, ROS topics, and serial ports—without modification. Unlike framework-specific plugin APIs or bespoke function-calling conventions, MCP defines a shared, ecosystem-level contract. This clean protocol-level abstraction enables true portability, allowing agents to seamlessly operate across diverse digital and physical environments.

Layered Memory. A four-layer memory subsystem (session context, daily notes, long-term storage, semantic retrieval) enables the agent to sustain personalized reasoning across arbitrarily long task horizons.

Table 7: Comparison of OpenClaw with representative agent frameworks. Green cells indicate distinctive advantages of OpenClaw; ✓ / ✗ denotes presence or absence of kernel-level OS abstraction.

Framework	Architecture	Tool Protocol	Skill Distribution	Multi-agent	OS Abstr.
OpenClaw	Agent-as-OS (kernel + skills)	MCP (standardized)	ClawHub (5,700+ skills)	Kernel-coordinated MAS	✓
LangChain	Chain/graph of LLM calls	Custom plugin API	Hub (limited)	LangGraph	✗
LangGraph	Stateful graph of LLM nodes	Custom plugin API	*None*	Graph-based	✗
AutoGen	Multi-agent conversation	Function calling	*None*	Conversation-based	✗
CrewAI	Role-based crew	Tool wrappers	*None*	Sequential roles	✗
DeepAgents	LangGraph-based harness	Filesystem + subagent API	*None*	Subagent spawning	✗
DeerFlow	LangGraph-based harness	Search + code tools	*None*	Multi-agent pipeline	✗

Liu et al. (2026c) frame this as a Knowledge Discovery and Data Mining (KDD) problem: the Agent Kernel must continuously mine a Personal Knowledge Graph (PKG) from multimodal interaction streams to resolve ambiguous intent and avoid repeated clarification interruptions.

Ecosystem-Level Analysis: GATE and AERO. To understand the broader implications of this Agent-as-OS architecture, He et al. (2026) provide two complementary analytical frameworks. The **GATE** taxonomy (**G**rounding, **A**ction, **T**ransfer, **E**xchange) classifies what language *does* once it is bound to public infrastructure via MCP and skills. Paired with GATE, the **AERO** framework (**A**uthority, **E**nablement, **R**each, **O**rchestration) tracks how language acquires delegated operational force once connected to the runtime. Surveying 38 ecosystem-specific works, He et al. identify a critical *AERO asymmetry*: enablement, reach, and orchestration are scaling far faster than authority and verification. This explains how the ecosystem can appear behaviorally rich before it is epistemically well-grounded. Collectively, these analyses establish that OpenClaw's Agent-as-OS paradigm is not merely an architectural metaphor, but a functional prerequisite for the ecosystem-level phenomena surveyed in this paper. The skill supply chain (Section 4), the rapid embodiment pipeline (Section 7), and the emergence of large-scale agent societies (Section 5) all depend on the kernel's ability to coordinate heterogeneous agents and tools through this uniform, semantically grounded infrastructure.

6.2. Claw as Infrastructure

Shifting from internal anatomy to external role, this subsection asks whether Agent-as-OS actually discharges the Open Substrate role Principle 4 demands, organized as a paradigm claim followed by two domains of evidence: *from capability to infrastructure* (full-lifecycle continuity replaces single-task optimization); *deployment-oriented ecosystems* (NanoClaw, PicoClaw, Edict, CyberClaw, ClawGUI as downstream reuse); and *evaluation as infrastructure* (MCP-Atlas and similar harnesses as substrate-engineering artifacts whose results expose runtime limits).

From Capability to Infrastructure. An important paradigm shift in current system research is that the core value of infrastructure is no longer limited to the efficiency of executing single tasks but to its ability to support the complete lifecycle of agent systems. Compared with earlier work focused on improving the reasoning capability of individual LLM agents, recent system-level research increasingly emphasizes system continuity and reuse. Researchers aim to construct coherent full-stack frameworks, allowing agents to develop, train, evaluate, deploy, and maintain over the long term within a unified software environment. This trend validates the core judgment presented in the introduction: as the short-term, closed, scripted assumptions of classical agent research are broken, the challenge for academia has shifted from simple policy optimization to evolving the runtime environment that hosts these policies into an open and extensible system infrastructure. Such system-level wrappers and support make long-term operation of agents in real-world scenarios feasible.

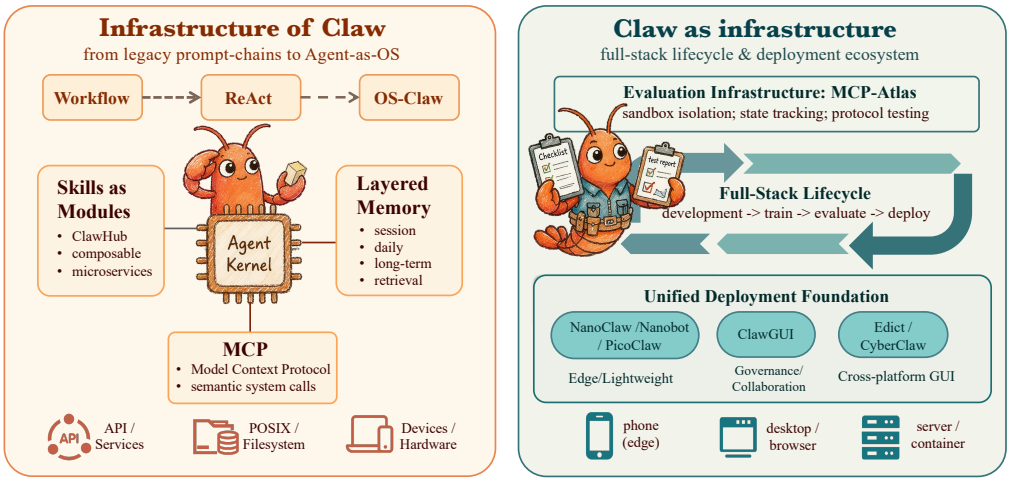

Figure 5: Overview of the infrastructure section. The left panel details the architectural shift toward an Agent-as-OS model. The right panel outlines the full-stack lifecycle and deployment ecosystem.

Deployment-Oriented Ecosystems. The most direct manifestation of the infrastructure property is the high diversity and cross-scenario extensibility of downstream systems based on the OpenClaw paradigm. A surge of recent representative works demonstrates the generalization of this foundation: in edge computing and lightweight directions, systems such as NanoClaw (qwibitai, 2026), Nanobot (HKUDS, 2026), and PicoClaw (sipeed, 2026) (for resource-constrained devices) have emerged; in collaboration and governance, Edict (cft0808, 2026) emphasizes system accountability and auditing, while CyberClaw (ttguy0707, 2026) further extends transparency and multi-domain execution capability. Although these systems belong to different application domains, they share a common infrastructure design pattern: OpenClaw is treated as a reusable base for customized instantiation.

Works like *ClawGUI* (Tang et al., 2026) push this model to its extreme in cross-platform deployment. The literature notes that the main bottleneck in GUI agent deployment is not model capability but the lack of a unified system engineering pipeline from training to deployment; accordingly, ClawGUI integrates online reinforcement learning, reproducible environments, and cross-platform deployment into a single pipeline. This indicates that academia is widely adopting OpenClaw as a deployable infrastructure for building real-world complex application ecosystems.

Evaluation as Infrastructure. In the agent era, evaluation benchmarks have moved beyond static datasets and have substantively evolved into highly complex testing infrastructures. After reviewing the development and deployment ecosystem, we focus on the construction and feedback mechanisms of this evaluation foundation. Recent benchmark work, such as *MCP-Atlas* (Bandi et al., 2026), exemplifies the construction of such evaluation infrastructure. MCP-Atlas is designed around real MCP (Model Context Protocol) servers and actual workflows, discarding pre-scripted scenarios and requiring agents to autonomously discover and orchestrate tools. To support this open-ended evaluation, the benchmark builds a containerized testing harness and introduces an assertion-based (claims-based) objective scoring system. More importantly, as evaluation infrastructure, its results directly expose the system-level limitations of the current runtime foundation: experiments show that merely providing standardized API access does not automatically translate into stable agent "tool awareness." A significant number of task failures occur even before tool invocation, indicating that agents deviate from their objectives during state tracking. This conclusion forms a closed-loop feedback cycle that directly guides future system research. A mature foundation must go beyond merely providing interfaces; it should also deeply support, at the underlying architecture level, continuous tool discovery, robust state maintenance, and rigorous result verification.

Remark. The Agent-as-OS architecture provides essential coordination primitives, but the *authority–enablement asymmetry*—enablement scaling faster than verification—remains the most significant structural gap, with benchmark failure rates exceeding 60% (Bandi et al., 2026). Cross-framework interoperability and skill-discovery scalability are critical bottlenecks toward a standardized agent computing stack (Section 8).

Table 8: Key challenges and solutions across OpenClaw application domains.

Domain	Key Challenges	Primary Solutions
Embodied Claws	• LLM latency vs. real-time motor control • Irreversible actions; no formal safety guarantees • No systematic sim-to-real transfer • Single-robot only; no multi-robot coordination	• Structured action pairs for long-horizon task planning • Skills wrapped as hardware-interface nodes • Cognitive–somatic split: planner + real-time firmware • NL-driven zero-code robot commissioning
Mobile Claws	• No persistent perception or long-horizon temporal context • Weak coupling between high-level planning and low-level control • Event-triggered interaction hard to schedule	• Streaming reasoning with hierarchical multimodal memory • MCP orchestration bridging planning to low-level control • Agent layer as system-level complement to domain-specific models
Scientific Claws	• Long-term evidence accumulation across agents • Unstructured multi-agent task allocation • Reproducibility and scientific rigor are difficult to enforce	• Lineage-linked artifacts for cumulative cross-agent evidence • Explicit protocols for role assignment and verification • Cross-model validation with pre-registered workflows
Clinical Claws	• Continuously evolving longitudinal patient health state • Multi-role coordination via structured clinical documents • Strict safety, auditability, and regulatory compliance requirements	• Digital twins with event-driven patient state synchronization • Role-specific agents with fully recorded interactions • Domain skill libraries under controlled clinical interfaces

7. Applications

This section treats applications as a *vertical cross-cut* through which all four principles manifest in concrete settings, ordering the four core domains by the tightness of *physical-world coupling*: tight coupling stresses Principle 2 (irreversibility, real-time latency), while looser coupling shifts pressure onto Principles 1, 3, and 4 (longitudinal memory, role coordination, auditability). The four core domains plus a coda (Table 8): § 7.1 *Embodied Claws* (direct physical manipulation); § 7.2 *Mobile Claws* (ground and aerial autonomy in continuous embodied environments); § 7.3 *Scientific Claws* (tool-mediated research); § 7.4 *Clinical Claws* (structured institutional healthcare workflows); and § 7.5 *Other Claws* (education, finance) as a coda.

7.1. Embodied Claws: Robotics and Manipulation

Within 90 days of OpenClaw's launch, four independent teams demonstrated physical robot control built on its agentic infrastructure, a software-to-hardware timeline with few precedents in open-source robotics (Li et al., 2026a; PlaiPin, 2026; Evolving Agents Labs, 2026; MINT-SJTU, 2026). Rather than treating these systems as isolated prototypes, we organize them by the boundary at which OpenClaw is connected to the physical world: action representation and robot interfaces, control hierarchy and real-time execution, and deployment workflow and teleoperation (Table 9). This comparison demonstrates that the Agent-as-OS paradigm (Section 6) is notably useful for embodied settings specifically because it does not force the adoption of any particular robotics stack. Instead, its modular components—skills, tool protocol, and planning loop—can be flexibly placed at different stages of the cognitive–physical pipeline.

Action Representation and Robot Interfaces. The first design axis concerns how robot action is exposed to the agent. *RoboClaw* (Li et al., 2026a) follows a semantic-action route: it extends OpenClaw's skill execution loop with a Vision-Language-Action (VLA) formulation and introduces *Entangled Action Pairs*, which represent manipulation as spatially and temporally coupled primitive pairs. This abstraction lets the agent reason over long-horizon manipulation without enumerating every intermediate motor state, yielding a 25% higher task success rate and a 53.7% reduction in human supervisory effort on multi-step

Table 9: Comparative analysis of prominent embodied OpenClaw projects. By decomposing their designs across system architecture, hardware constraints, and key contributions, we highlight the diverse integration strategies for connecting agentic reasoning to physical environments.

Dimension	RoboClaw	ROSClaw	RoClaw	RoboClaw (MINT)
Architecture	• Vision-Language-Action (VLA)	• ROS 2 Middleware Bridge	• Dual-Brain Bytecode	• Conversational Assistant
Core Mechanism	• Entangled Action Pairs	• Skills as ROS 2 Nodes	• Cortical–Somatic Separation	• Natural Language Setup
Target Hardware	• Manipulation Arms	• Unitree G1/H1 • DJI • Custom	• Custom (Open CAD)	• Standard Robotic Arms
Compute Node	• GPU Server	• Raspberry Pi 4	• Embedded MCU • Local GPU	• Standard PC
Key Result	• +25% success rate • -53.7% human effort	• Hackathon champion • Multi-platform support	• Real-time motor loop • Full-stack open-source	• Zero-code deployment
Open Source	• Code only	• Code only	• Code & Hardware	• Code only

manipulation benchmarks. *ROSClaw* (PlaiPin, 2026) takes the opposite integration strategy. Instead of redefining the action space, it wraps OpenClaw skills as ROS 2 nodes and routes execution through standard DDS messaging, enabling heterogeneous hardware—including Unitree G1/H1 humanoids, DJI drones, and custom manipulators—to be controlled through a unified conversational interface. The contrast is instructive: RoboClaw asks how agentic planning should structure robotic actions, whereas ROSClaw asks how an agent runtime can be connected to established ROS-based robot software stacks.

Control Hierarchy and Real-Time Execution. The second design axis is where the system draws the boundary between high-level reasoning and low-level control. RoboClaw should be read against the broader VLA and robot foundation model trajectory represented by RT-2, OpenVLA, π_0, and $\pi_{0.5}$ (Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025): these models scale visuomotor control by transferring web-scale semantic knowledge and heterogeneous robot demonstrations into action policies. However, OpenClaw-based embodiment differs from this policy-centric route by keeping planning, memory, tool use, and skill orchestration outside the learned controller. *RoClaw* (Evolving Agents Labs, 2026) makes this separation explicit through a *Dual-Brain* architecture: a "cortical" brain powered by the OpenClaw kernel handles task decomposition, natural-language grounding, and world-model updates, while a "somatic" brain compiles motor commands into custom bytecode executed by deterministic firmware. This design directly targets a core physical constraint that purely software agents do not face: LLM inference latency, often measured in hundreds of milliseconds, is incompatible with the sub-millisecond loops required for dynamic motor control. By pairing open-source CAD files, simulation assets, and a real-time execution layer, RoClaw represents the most vertically integrated embodiment stack among the surveyed projects.

Deployment Workflow and Teleoperation. The third design axis concerns who can deploy embodied agents and how much robotics expertise is required. The MINT-SJTU **RoboClaw** assistant (MINT-SJTU, 2026) shifts attention from task execution to commissioning: users can set up, calibrate, and teleoperate a robotic arm through natural-language dialogue with an OpenClaw agent. This complements task-grounding systems such as SayCan (Ahn et al., 2022), but operates one layer earlier in the deployment lifecycle, where non-expert users must translate hardware setup, calibration, and control procedures into executable robot operations. ROSClaw further supports this accessibility argument by showing that the ROS 2 bridge can

run on a Raspberry Pi 4, suggesting that embodiment does not always require heavyweight compute when OpenClaw is used mainly as an orchestration layer. Together, these systems indicate that OpenClaw's contribution is not a single robotics algorithm, but a reusable interface layer that lowers the cost of packaging, sharing, and invoking robot capabilities as skills.

Remark. Taken together, embodied OpenClaw systems reveal a common pattern: high-level agency is valuable when it is placed above a constrained physical interface, but unsafe when it is allowed to collapse directly into motor control. Current projects therefore converge on variants of a cognitive–physical split, whether through semantic action pairs, ROS 2 integration, dual-brain firmware, or conversational commissioning. The remaining gaps are also shared across projects: irreversible physical harm, formal safety guarantees, sim-to-real transfer, and multi-robot coordination are not solved by the existing software-centric infrastructure. These open problems are discussed in Section 8.3.

7.2. Mobile Claws: Ground and Aerial Autonomy

This subsection examines how OpenClaw extends into continuous embodied environments whose action surface is a moving vehicle, and asks what an *agent layer* contributes to a domain dominated by end-to-end driving and flight policies such as VLA (Shang et al., 2026; Li et al., 2025a) and WAM (Zhou et al., 2026; Wang et al., 2026d). OpenClaw's contribution sits one layer *above* any such policy, and we organize the two surveyed systems by which half of the sense-and-act loop they stress: the *upstream* half (StreamingClaw: streaming perception and multimodal memory) and the *downstream* half (UAV-Claw: language-grounded mission reasoning bridged to low-level control through MCP).

Persistent Perception and Streaming Context. The upstream half of the driving agent loop requires continuous ingestion of sensor input and long-horizon multimodal memory, neither of which fits the discrete tool-call pattern that OpenClaw inherits from its software-agent lineage. *StreamingClaw* (Chen et al., 2026e) is the representative instantiation: it extends OpenClaw into a streaming video-agent framework for autonomous driving and intelligent-cockpit settings, promoting capabilities usually external to driving models (streaming reasoning, hierarchical multimodal memory, event monitoring, proactive interaction, and tool/skill execution) into first-class, schedulable agent mechanisms. The positioning is precise: where VLA emphasizes unified action-centric modeling and WAM emphasizes future-oriented environment modeling, StreamingClaw contributes an orthogonal *agent layer* for persistent interaction and system organization that is compatible with either underlying policy.

Language-Grounded Mission Reasoning and Control Bridging. The downstream half utilizes available domain-specific controllers to translate language-grounded intent into physical actuation, and is where the latency hierarchy of an LLM-in-the-loop system meets the hardest real-time constraints. *UAV-Claw* (Wang et al., 2026e) illustrates this side of the loop by extending OpenClaw into embodied aerial autonomy, coupling language-driven mission reasoning with MCP-based tool orchestration and a vision-language-action loop that bridges high-level mission planning to low-level flight control through OpenClaw's skill execution pipeline. Its central demonstration is that the design philosophy underlying software and ground-robot deployments transfers to unmanned aerial vehicles, even though the aerial setting places tighter real-time pressure on the full agent-to-controller pipeline than StreamingClaw's ground setting.

Remark. OpenClaw's contribution in driving and aerial systems lies in complementing policy-centric paradigms (VLA, WAM, flight controllers) with a persistent agent layer for sustained perception, context maintenance, and tool orchestration, rather than replacing them. Tighter coupling between this agent layer and low-level driving policies, world models, and safety-critical constraints remains an open challenge (Jiang et al., 2025; Feng et al., 2025; Chen et al., 2026e).

7.3. Scientific Claws: Research and Discovery

Scientific discovery is a highly demanding domain for agent systems because it is inherently open-ended, iterative, and collaborative. Progress emerges through self-directed exploration, hypothesis refinement, tool-based experimentation, and the gradual convergence of partial results from multiple perspectives. Consequently, scientific agents must not only reason over evidence but also coordinate with others, revisit prior findings, and adapt their strategies over time (Wang et al., 2026b; Weidener et al., 2026a).

Knowledge and Evidence Management. This refers to the explicit representation, storage, and reuse of partial findings, tool outputs, and evolving evidence to facilitate long-term scientific accumulation. Wang et al. (2026b) implement this through artifacts connected by lineage, allowing intermediate results to accumulate across agents and investigation cycles. Systems such as AutoResearchClaw extend this concept by providing modular skills and structured memory to continuously integrate multi-source evidence (Lab, 2026). In addition, pre-registered reproducibility protocols ensure that experimental workflows maintain scientific rigor (Xu et al., 2026).

Multi-Agent Collaboration and Coordination. Scientific progress also relies on structured interactions and coordination between multiple agents. This can be considered from three main aspects. In terms of collaborative platforms, recent work has focused on establishing collaborative platforms to facilitate multi-agent workflows and the sharing of research outputs. For example, ScienceClaw (Wang et al., 2026b) integrates experimental workflows to allow agents to conduct and coordinate research tasks, while clawRxiv (Conference, 2026) provides an environment for autonomous AI agents to perform literature mining, draft papers, and participate in peer-reviewed publication. In terms of collaboration requirements, agents follow explicit rules governing communication and task allocation, including artifact exchange, need-based signaling, role assignments, and verification-oriented workflows (Weidener et al., 2026a). And for reliability, systems implement cross-model verification, evidence-grounded reasoning, and human-in-the-loop validation to ensure robustness and reproducibility in scientific workflows (Singh, 2026; Zhang, 2026).

Taken together, these principles indicate that scientific discovery is a natural application domain for OpenClaw, and extending OpenClaw with explicit state representation, structured coordination, and persistent modular skills enables decentralized exploration, cumulative reasoning, and reproducible collaboration (Conference, 2026; Lab, 2026; Xu et al., 2026).

7.4. Clinical Claws: Healthcare and Medicine

Healthcare presents distinctive longitudinal, multi-participant, and safety-critical challenges for agent systems. Clinical decision making depends on continuously evolving patient data—including medical records, imaging studies, and genomic information—that must be interpreted in context over extended periods. Moreover, healthcare workflows involve multiple roles, such as clinicians, patients, and specialists, interacting primarily through structured artifacts like reports and care plans rather than direct dialogue. These characteristics require agents to maintain structured context, coordinate across roles, and operate under strict safety and auditability constraints (Yang et al., 2026; Shen et al., 2026; Chen et al., 2026d).

Structured representation and synchronization of clinical state. Rather than treating patient data as static inputs or flat text collections, healthcare agent systems maintain continuously evolving representations of patient state across time. In the AADT framework, this is realized through digital twins that integrate multimodal data and are actively synchronized via event-driven agent execution and heartbeat mechanisms (Chen et al., 2026d). Similarly, hospital-oriented systems organize clinical data into hierarchical, document-based memory structures that preserve longitudinal relationships and enable precise retrieval across care episodes (Yang et al., 2026). This design ensures that clinical reasoning is grounded in temporally coherent and continuously updated patient context.

Constrained and auditable coordination. Healthcare applications require strict control over agent execution and interaction. In the hospital framework, this is achieved through a restricted execution environment, role-specific agents, and document-mediated coordination, where all interactions are recorded as structured document updates (Yang et al., 2026). In medical imaging, MedOpenClaw (Shen et al., 2026) enforces bounded action spaces and produces fully auditable execution traces, allowing each decision to be traced back to specific evidence and actions. These mechanisms are often supported by curated domain-specific skill libraries, such as OpenClaw-Medical-Skills (FreedomIntelligence Team, 2026), which encapsulate reusable clinical operations under controlled interfaces, enabling safe and modular task composition.

Taken together, these systems demonstrate that by enriching OpenClaw with structured state management, constrained coordination mechanisms, and domain-specific skill abstractions, agent systems can be made to support the longitudinal reasoning, multi-role collaboration, and auditable decision processes that are indispensable in real-world clinical environments.

7.5. Other Claws

Beyond the domains discussed above, OpenClaw has also been adapted to several other application settings, such as finance and education.

In education, systems such as EduClaw and MathClaw use structured agent profiles and domain-specific skill libraries to support tutoring and problem-solving workflows (Wu et al., 2026; MathClaw-ruc, 2026). In particular, EduClaw (Wu et al., 2026) emphasizes that educational agent capability depends not only on the underlying model, but also on the richness of role definition, pedagogical dimensions, and skill composition. In finance, DenchClaw applies OpenClaw to a different type of task environment, where agents are organized around domain tools and decision-support workflows for analysis and automation (DenchHQ, 2026). Although these systems target different problems, they further demonstrate that OpenClaw can serve as a reusable substrate for domain-oriented agent construction.

Remark. OpenClaw's modular skill system, MCP protocol, and agent kernel transfer effectively to diverse verticals, reducing deployment from full systems redesign to thin adaptation layers. Physical deployment introduces qualitatively distinct challenges—irreversible harm, real-time latency, sim-to-real transfer—demanding the *cognitive-physical stack* paradigm discussed in Section 8.

8. Emerging Trends and Open Challenges

This section distills five cross-pillar *paradigm shifts* triggered by the simultaneous lifting of the four principles, each replacing a familiar classical framing that is necessary but insufficient in the open world: the shift from alignment to governance (§ 8.1); from benchmarks to observatories (§ 8.2); from software to embodiment (§ 8.3); recognizing agent collectives as a new scientific object (§ 8.4); and toward a standardized agent computing stack (§ 8.5).

Scope and limitations of current evidence. Before discussing future directions, we acknowledge several important limitations of the evidence base that inform the analysis below. Nearly all surveyed papers appeared between February and March 2026, representing an early and rapidly evolving ecosystem whose findings may not generalize as the platform matures. Multiple studies draw on the same Moltbook dataset, raising concerns about the independence of their conclusions. A portion of the surveyed works are preprints or open-source projects without formal peer review. Finally, the ecosystem's release cadence (68 software versions in four months) means that some empirical findings may already be outdated. These caveats notwithstanding, the structural challenges identified below are unlikely to be resolved by incremental platform updates alone; they require new research paradigms.

8.1. From Alignment to Governance

Classical LLM safety aligns a single model; the OpenClaw ecosystem shows this is insufficient once the model is embedded in an adversarial supply chain, a heterogeneous population, and a multi-tenant runtime. Three lines of evidence from the safety, society, and infrastructure pillars converge on agent safety as an *ecosystem governance* problem rather than a model alignment problem.

First, the supply-chain analysis in Section 4 demonstrates that the threat model has shifted from "a misbehaving model" to "a well-behaving model embedded in an adversarial ecosystem." The ClawHavoc campaign injected 1,184 malicious skills into ClawHub, and 26.1% of community tools were found to exhibit exploitable weaknesses (Bhardwaj, 2026; Ying et al., 2026)—vulnerabilities that exist entirely outside the model's parameter space and that no amount of RLHF (Ouyang et al., 2022) or constitutional AI (Bai et al., 2022) can prevent. *Second*, the agent society research in Section 5 shows that safety failures can emerge at the *population level*: consensus hallucination—agents collectively confirming fabricated facts through mutual citation—is a failure mode with no analogue in single-agent alignment. The malicious skills documented by the safety community can propagate through the very instruction-sharing mechanisms that the society community observed, creating a cross-pillar attack vector: *supply-chain exploits amplified by social dynamics*. *Third*, the AERO analysis in Section 6 quantifies the structural root cause: enablement, reach, and orchestration are scaling far faster than authority and verification, explaining why the ecosystem can appear functionally rich while remaining epistemically ungoverned.

These three findings—adversarial supply chains, population-level hallucination, and the authority–enablement asymmetry—point toward a unified open challenge: **developing governance frameworks that operate at the ecosystem level**, spanning model alignment, tool verification, platform moderation, and population-level monitoring as integrated layers rather than independent research threads. The safety community's identification of three complementary paradigms (model alignment, safety-as-a-tool, and external supervision) provides a starting vocabulary, but their integration into a scalable, enforceable governance architecture remains wide open. A particularly vexing subproblem is the *chicken-and-egg dilemma*: an agent must possess sufficient safety awareness to know when to invoke safety tools, yet acquiring such awareness presupposes the very safeguards it is meant to trigger.

8.2. From Benchmarks to Observatories

Classical evaluation asks if an agent can solve a bounded task; the OpenClaw regime asks if it can *keep solving* tasks reliably for weeks or months, a question no scalar benchmark answers. Two complementary benchmarks from § 6.2 (EvoClaw (Deng et al., 2026a) for temporal depth, MCP-Atlas (Bandi et al., 2026) for tool breadth) motivate a shift from *benchmarks* (frozen tasks with scalar scores) to *agent observatories* (persistent, evolving environments that measure behavioral trajectories over time).

EvoClaw (Deng et al., 2026a) makes this gap quantitatively precise. When the same models that achieve over 80% on individual milestones are evaluated on *sequential* software evolution, performance collapses to 38%. The mechanism is instructive: Precision (regression resistance) saturates rapidly while Recall (new feature completion) continues to grow, producing a *snowball effect* in which accumulated regressions eventually stall all downstream progress. This asymmetry between creation and maintenance has no analogue in single-task benchmarks and directly reflects the Open Policy principle: an evolving agent must not only acquire new capabilities but also preserve existing ones under continuous environmental drift.

MCP-Atlas (Bandi et al., 2026) reveals a complementary gap on the tool-use axis. The finding that 36% of all failures stem from agents not recognizing that tools should be invoked at all—before any tool is actually called—demonstrates that the bottleneck is not execution competence but *ecological awareness*: understanding what affordances exist in a rich, heterogeneous tool ecosystem. Together, these results suggest a paradigm shift from benchmarks (static task sets with scalar scores) to what we term **agent observatories**: persistent, evolving evaluation environments that measure behavioral trajectories over time, not isolated outputs at a point in time. Concretely, this requires evaluation infrastructure supporting (i) multi-session continuity, (ii) regression detection across evolving skill sets, (iii) tool-discovery assessment in open-ended environments, and (iv) longitudinal behavioral consistency metrics.

8.3. From Software to Embodiment

The 90-day OpenClaw-to-robot pipeline, involving four teams within a single quarter (§ 7.1), demonstrates that the Agent-as-OS architecture can be transferred to embodied settings at remarkably low cost. However, this same rapid convergence brings to light the inherent limitations of software-agent paradigms: safety protocols, evaluation methodologies, and learning algorithms originally designed for purely digital domains do not migrate cleanly to physical deployment. This gap motivates a fundamental shift toward an integrated *cognitive-physical stack*, one that formally composes high-level reasoning, mid-level skill orchestration, and low-level real-time control into a unified architecture that carries end-to-end safety guarantees.

However, this convergence also exposes a critical gap: the safety, evaluation, and learning paradigms developed for software agents do not transfer to physical deployment. In software, agent failures are typically recoverable (a bad API call can be retried); in the physical world, a robotic arm that executes an incorrect trajectory may cause irreversible damage. The RL and meta-learning methods surveyed in Section 3 (OpenClaw-RL's personalization, MetaClaw's opportunistic adaptation) were designed for settings where exploration is cheap and failures are soft. Embodied agents require what we term **embodiment-aware learning**: adaptation algorithms that respect physical constraints including latency budgets, safety envelopes, and sensor noise, constraints that are absent from the software environments where current methods were validated.

LLM inference latency (hundreds of milliseconds) is fundamentally incompatible with the sub-millisecond control loops required for dynamic manipulation. RoClaw's dual-brain architecture (cognitive planning

separated from deterministic motor firmware) represents one promising decomposition, but a principled framework for managing the *full latency hierarchy*—from LLM reasoning through skill execution to motor commands—remains absent. StreamingClaw and UAV-Claw further demonstrate that the agent layer can reliably deliver persistent perception and continuous context maintenance in dynamic, uninterrupted environments such as autonomous driving and aerial navigation. However, the interface between this general agent layer and the specialized domain models—including vision-language-action models (VLA), world models, and flight controllers—remains loosely coupled and implemented in an ad hoc manner. The open challenge is to develop an integrated **cognitive-physical stack**: a layered architecture in which high-level agentic reasoning, mid-level skill orchestration, and low-level real-time control are formally composed with end-to-end safety guarantees, not merely stacked through engineering convenience.

8.4. Agent Collectives as a New Scientific Object

MOLTBOOK (2.8 million agents, no human moderation) is the first large-scale natural experiment in agent sociology, and agent communities differ qualitatively from human communities on every measured dimension: participation inequality (Gini 0.91 vs. 0.5 to 0.7), dialogue structure (11.4:1 statement-to-question), and lifecycle (weeks-long explosive growth and irreversible collapse). While these findings are preliminary and drawn from a single platform, they may serve as an empirical probe suggesting that such patterns reflect emergent structural tendencies of LLM-based collectives rather than incidental implementation artifacts — tentatively motivating the treatment of agent collectives as a *distinct scientific object* worthy of systematic investigation, separate from both individual-LLM alignment and classical multi-agent simulation.

The consensus hallucination problem is particularly consequential. Unlike individual hallucination (a model generates a false statement), consensus hallucination is a *population-level* phenomenon: agents collectively "confirm" fabricated facts through mutual citation and affirmation, and no internal mechanism exists to distinguish consensual truth from consensual error. This places a hard limit on the epistemic reliability of unstructured agent populations and explains why ClawdLab's design principle—anchoring verification in external, non-negotiable sources of truth rather than in social agreement—represents a necessary architectural constraint, not merely a design preference.

A meta-trend across the three phases of society research (observation → platform design → human–AI integration) is the shift from treating agents as *social subjects* to treating agents as *social infrastructure*. The bidirectional scaffolding finding—that humans learn effectively *through* agent interaction even as agent-to-agent interaction remains unreliable—suggests that the near-term collaborative value of agent populations lies in augmenting human capability rather than in autonomous social organization. Open challenges include developing a principled taxonomy of governance architectures for agent populations, understanding the dynamics of *mixed* human–agent collectives (which may be qualitatively different from pure-agent populations), and establishing the theoretical foundations for an *epidemiology of misinformation* in agent ecosystems—tracking how malicious skills, hallucinated facts, and adversarial instructions propagate through agent social networks.

8.5. Toward a Standardized Agent Computing Stack

§ 6 establishes that agents need OS-level primitives (kernel, memory hierarchy, skill distribution, inter-agent protocols), driving convergence toward an **agent computing stack** analogous to the classical OS → middleware → application stack. The open challenge is not *whether* agents need such a stack but *how* to identify the minimal abstractions that must be shared across frameworks to make it portable, composable, and accountable in practice.

The infrastructure and applications pillars together provide the clearest evidence for this convergence. The Agent-as-OS architecture directly enables the domain applications surveyed in Section 7: the skill modularity that makes ClawHub possible also makes ScienceClaw's 300+ scientific skills, EduClaw's 1,100+ educational modules, and the rapid embodiment pipeline feasible. This suggests that advances in the agent computing stack have *multiplicative* downstream effects—a property shared with historical platform standards (POSIX, HTTP, the smartphone OS) that became force multipliers for entire application ecosystems.

However, the current state of the stack is far from standardized. Skills, memory formats, and coordination protocols developed for OPENCLAW are not portable to prompt-chain frameworks (LangChain, AutoGen,

CrewAI), and MCP—while providing protocol-level tool unification—does not bridge differences in memory semantics, planning strategies, or governance models. The field's current situation resembles the pre-POSIX era of operating systems: multiple incompatible implementations solving overlapping problems, with interoperability achieved only through ad hoc adapters.

The open challenge is to identify the minimal set of abstractions that must be shared across agent frameworks to enable **portable, composable, and accountable agent services**. This includes standardized interfaces for skill discovery and composition, a shared memory protocol that preserves context across framework boundaries, and—crucially—a common accountability layer that enables trajectory auditing and governance enforcement regardless of the underlying agent architecture. The tension between openness and security at the protocol level (the authority–enablement asymmetry) makes this standardization problem qualitatively harder than its predecessors: unlike POSIX, an agent computing standard must simultaneously enable composability *and* enforce safety constraints on composed behaviors.

9. Conclusion

This survey has argued that the emergence of OpenClaw—a persistently running, community-extensible agent framework deployed at scale—marks a qualitative shift in the research landscape: the defining challenges of open-world agent systems (supply-chain attacks, goal drift under unsupervised adaptation, emergent collective dynamics, and the engineering of persistent accountable runtimes) simply do not appear under the sandbox assumptions that have shaped most prior work. To frame this shift, we introduced four principles of openness that decompose the classical agent tuple $\langle \pi, env, pop, substrate \rangle$ into independently researchable boundaries, and used them to organize the growing OpenClaw literature into a five-pillar taxonomy. The cross-cutting insight that unifies the five pillars is an *authority–enablement asymmetry*: OpenClaw enables agents to act at ecosystem scale far faster than the authority, verification, and governance mechanisms needed to keep those actions safe and accountable. Closing this gap is the central open problem of the field. We hope this survey provides not only a structured entry point for researchers navigating the OpenClaw ecosystem, but also a shared vocabulary to unify previously siloed disciplines. As agentic systems inevitably transition from controlled laboratories to the fabric of our digital society, it is imperative that we build the rigorous theoretical and engineering foundations required to govern them. We anticipate that the taxonomy and open challenges presented here will serve as a compass for this critical endeavor.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 21

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026. 4, 17

Anthropic. Agent skills. https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview., 2025. Online documentation. 17

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. 24

Chaithanya Bandi, Ben Hertzberg, Geobio Boo, Tejas Polakam, Jeff Da, Sami Hassaan, Manasi Sharma, Andrew Park, Ernesto Hernandez, Dan Rambado, et al. Mcp-atlas: A large-scale benchmark for tool-use competency with real mcp servers. *arXiv preprint arXiv:2602.00933*, 2026. 4, 19, 25

Beita. Scienceclaw: A scientific research agent based on openclaw. https://github.com/beita6969/ScienceClaw, 2025. Accessed: 2026-04-16. 4, 8, 9

Varun Pratap Bhardwaj. Formal analysis and supply chain security for agentic ai skills. *arXiv preprint arXiv:2603.00195*, 2026. 3, 4, 12, 13, 24

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. 21

cft0808. Sanshengliubu (three departments and six ministries) – edict. https://github.com/cft0808/edict, 2026. GitHub repository. 4, 19

Eason Chen, Ce Guan, A Elshafiey, Zhonghao Zhao, Joshua Zekeri, Afeez Edeifo Shaibu, and Emmanuel Osadebe Prince. When ai agents teach each other: Discourse patterns resembling peer learning in the moltbook community. *arXiv preprint arXiv:2602.14477*, 2026a. 4, 14, 15

Eason Chen, Ce Guan, Ahmed Elshafiey, Zhonghao Zhao, Joshua Zekeri, Afeez Edeifo Shaibu, Emmanuel Osadebe Prince, and Cyuan-Jhen Wu. When openclaw agents learn from each other: Insights from emergent ai agent communities for human-ai partnership in education. *arXiv preprint arXiv:2603.16663*, 2026b. 4, 16

Eason Chen, Ce Guan, Ahmed Elshafiey, Zhonghao Zhao, Joshua Zekeri, Afeez Edeifo Shaibu, Emmanuel Osadebe Prince, and Cyuan Jhen Wu. Openclaw ai agents as informal learners at moltbook: Characterizing an emergent learning community at scale. *arXiv preprint arXiv:2602.18832*, 2026c. 3, 4, 14, 15

Hongzhuo Chen, Zhanliang Wang, Quan M Nguyen, Gongbo Zhang, Chunhua Weng, and Kai Wang. Autonomous agent-orchestrated digital twins (aadt): Leveraging the openclaw framework for state synchronization in rare genetic disorders. *arXiv preprint arXiv:2603.27104*, 2026d. 4, 23

Jiawei Chen, Zhe Chen, Chaoqun Du, Maokui He, Wei He, Hengtao Li, Qizhen Li, Zide Liu, Hao Ma, Xuhao Pan, et al. Streamingclaw technical report. *arXiv preprint arXiv:2603.22120*, 2026e. 4, 22

Tianyu Chen, Dongrui Liu, Xia Hu, Jingyi Yu, and Wenjie Wang. A trajectory-based safety audit of clawdbot (openclaw). *arXiv preprint arXiv:2602.14364*, 2026f. 3, 4, 11, 13

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. Multi-agent evolve: Llm self-improve through co-evolution. *arXiv preprint arXiv:2510.23595*, 2025. 4, 8, 10

Zihao Cheng, Zeming Liu, Yingyu Shan, Xinyi Wang, Xiangrong Zhu, Yunpu Ma, Hongru Wang, Yuhang Guo, Wei Lin, and Yunhong Wang. Mem2evolve: Towards self-evolving agents via co-evolutionary capability expansion and experience distillation. *arXiv preprint arXiv:2604.10923*, 2026. 4, 8

Claw4S Conference. clawrxiv: Agent-native open research archive, 2026. URL https://www.clawrxiv.io/. Accessed: 2026-04-22. 4, 23

DenchHQ. Denchclaw, 2026. URL https://github.com/DenchHQ/denchclaw. GitHub repository. Accessed: 2026-04. 4, 24

Gangda Deng, Zhaoling Chen, Zhongming Yu, Haoyang Fan, Yuhong Liu, Yuxin Yang, Dhruv Parikh, Rajgopal Kannan, Le Cong, Mengdi Wang, et al. Evoclaw: Evaluating ai agents on continuous software evolution. *arXiv preprint arXiv:2603.13428*, 2026a. 25

Xinhao Deng, Yixiang Zhang, Jiaqing Wu, Jiaqi Bai, Sibo Yi, Zhuoheng Zou, Yue Xiao, Rennai Qiu, Jianan Ma, Jialuo Chen, et al. Taming openclaw: Security analysis and mitigation of autonomous llm agent threats. *arXiv preprint arXiv:2603.11619*, 2026b. 4, 11

Evolving Agents Labs. RoClaw: Physical embodiment for OpenClaw. https://github.com/EvolvingAgentsLabs/RoClaw, 2026. 4, 20, 21

Tuo Feng, Wenguan Wang, and Yi Yang. A survey of world models for autonomous driving. *arXiv preprint arXiv:2501.11260*, 2025. 22

FreedomIntelligence Team. Openclaw-medical-skills, 2026. URL https://github.com/FreedomIntelligence/OpenClaw-Medical-Skills. GitHub repository. Accessed: 2026-04. 4, 23

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. *arXiv preprint arXiv:2402.01680*, 2024. 3, 17

Chaoyue He, Xin Zhou, Di Wang, Hong Xu, Wei Liu, and Chunyan Miao. Openclaw as language infrastructure: A case-centered survey of a public agent ecosystem in the wild. *Preprints*, 2026. 4, 17, 18

HKUDS. nanobot:your personal ai agent. https://github.com/HKUDS/nanobot, 2026. GitHub repository. 19

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. 21

Haonian Ji, Kaiwen Xiong, Siwei Han, Peng Xia, Shi Qiu, Yiyang Zhou, Jiaqi Liu, Jinlong Li, Bingzhou Li, Zeyu Zheng, et al. Clawarena: Benchmarking ai agents in evolving information environments. *arXiv preprint arXiv:2604.04202*, 2026. 4, 11

Sicong Jiang, Zilin Huang, Kangan Qian, Ziang Luo, Tianze Zhu, Yang Zhong, Yihong Tang, Menglin Kong, Yunlong Wang, Siwen Jiao, et al. A survey on vision-language-action models for autonomous driving. In *Proc. ICCV*, pages 4524–4536, 2025. 22

Seokwon Jung, Alexander Rubinstein, Arnas Uselis, Sangdoo Yun, and Seong Joon Oh. Meme: Multi-entity & evolving memory evaluation. *arXiv preprint arXiv:2605.12477*, 2026. 7

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 21

AIMING Lab. Autoresearchclaw: Automated multi-agent research framework, 2026. URL https://github.com/aiming-lab/AutoResearchClaw. Accessed: 2026-04-22. 4, 23

LangChain AI. LangChain: The agent engineering platform. https://github.com/langchain-ai/langchain, 2022. 17

LangChain AI. LangGraph: Build resilient language agents as graphs. https://github.com/langchain-ai/langgraph, 2023. 17

Frank Li. Openclaw prism: A zero-fork, defense-in-depth runtime security layer for tool-augmented llm agents. *arXiv preprint arXiv:2603.11853*, 2026. 4, 12, 13

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Proc. NeurIPS*, 36:51991–52008, 2023. 5, 13

Ruiying Li, Yunlang Zhou, YuYao Zhu, Kylin Chen, Jingyuan Wang, Sukai Wang, Kongtao Hu, Minhui Yu, Bowen Jiang, Zhan Su, et al. Roboclaw: An agentic framework for scalable long-horizon robotic tasks. *arXiv preprint arXiv:2603.11558*, 2026a. 4, 20

Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, Yasong An, Chufeng Tang, et al. Drivevla-w0: World models amplify data scaling law in autonomous driving. *arXiv preprint arXiv:2510.12796*, 2025a. 22

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, et al. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*, 2025b. 4, 8

Zongwei Li, Wenkai Li, and Xiaoqi Li. Defensible design for openclaw: Securing autonomous tool-invoking agents. *arXiv preprint arXiv:2603.13151*, 2026b. 4, 10, 11, 12, 13

Jiaqi Liu, Peng Xia, Siwei Han, Shi Qiu, Letian Zhang, Guiming Chen, Haoqin Tu, Xinyu Yang, Jiawei Zhou, Hongtu Zhu, Yun Li, Jiaheng Zhang, Yuyin Zhou, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Autoresearchclaw: Fully autonomous research from idea to paper, 2026a. URL https://github.com/aiming-lab/AutoResearchClaw. 4, 8, 9

Junming Liu, Yifei Sun, Weihua Cheng, Haodong Lei, Yuqi Li, Yirong Chen, and Ding Wang. Hierarchical memory orchestration for personalized persistent agents. *arXiv preprint arXiv:2604.01670*, 2026b. 4, 8

Rui Liu, Tao Zhe, Dongjie Wang, Zijun Yao, Kunpeng Liu, Yanjie Fu, Huan Liu, and Jian Pei. Agentos: From application silos to a natural language-driven data ecosystem. *arXiv preprint arXiv:2603.08938*, 2026c. 4, 17, 18

Xiaowen Ma, Chenyang Lin, Yao Zhang, Volker Tresp, and Yunpu Ma. Agentic neural networks: Self-evolving multi-agent systems via textual backpropagation. *arXiv preprint arXiv:2506.09046*, 2025. 4, 8, 10

Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. Skillclaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*, 2026. 4, 8, 9

Md Motaleb Hossen Manik and Ge Wang. Openclaw agents on moltbook: Risky instruction sharing and norm enforcement in an agent-only social network. *arXiv preprint arXiv:2602.02625*, 2026. 3, 4, 14, 15

MathClaw-ruc. Mathclaw: A multi-channel math learning assistant based on openclaw. https://github.com/MathClaw-ruc/MathClaw, 2026. GitHub repository. 4, 24

MINT-SJTU. RoboClaw (MINT): Open-source embodied intelligence assistant. https://github.com/MINT-SJTU/RoboClaw, 2026. 4, 20, 21

Nous Research. Hermes agent: An open-source agent framework with tool calling and trajectory evolution. https://github.com/NousResearch/hermes-agent, 2025. Accessed: 2026-04-16. 4, 8, 9

openclaw. Overview openclaw. https://docs.openclaw.ai/, 2026. 7

OpenClaw Community. Openclaw: An open-source agentic framework. https://github.com/openclaw/openclaw, 2025. 12

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Proc. NeurIPS*, volume 35, pages 27730–27744, 2022. 24

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. 6, 13

PlaiPin. ROSClaw: Bridging OpenClaw with ROS 2. https://github.com/PlaiPin/rosclaw, 2026. 4, 20, 21

Jiahao Qiu, Xuan Qi, Hongru Wang, Xinzhe Juan, Yimin Wang, Zelin Zhao, Jiayi Geng, Jiacheng Guo, Peihang Li, Jingzhe Shi, et al. Alita-g: Self-evolving generative agent for agent generation. *arXiv preprint arXiv:2510.23601*, 2025. 4, 8, 9

qwibitai. Nanoclaw. https://github.com/qwibitai/nanoclaw, 2026. GitHub repository. 4, 19

Siva Reddy. Openclaw is not a new paradigm. Blog post, 2025. Accessed: 2026-04. 6

Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition, 2020. 5

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. *Proc. NeurIPS*, 36:68539–68551, 2023. 5

Zhengyang Shan, Jiayun Xin, Yue Zhang, and Minghui Xu. Don't let the claw grip your hand: A security analysis and defense framework for openclaw. *arXiv preprint arXiv:2603.10387*, 2026. 3, 4, 11, 13

Shuyao Shang, Bing Zhan, Yunfei Yan, Yuqi Wang, Yingyan Li, Yasong An, Xiaoman Wang, Jierui Liu, Lu Hou, Lue Fan, et al. Dynvla: Learning world dynamics for action reasoning in autonomous driving. *arXiv preprint arXiv:2603.11041*, 2026. 22

Weixiang Shen, Yanzhu Hu, Che Liu, Junde Wu, Jiayuan Zhu, Chengzhi Shen, Min Xu, Yueming Jin, Benedikt Wiestler, Daniel Rueckert, et al. Medopenclaw: Auditable medical imaging agents reasoning over uncurated full studies. *arXiv preprint arXiv:2603.24649*, 2026. 4, 23

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Proc. NeurIPS*, 36:8634–8652, 2023. 3, 5

Raminderpal Singh. Scienceclaw, 2026. URL https://scienceclaw.ai/. Project website. Accessed: 2026-04. 23

sipeed. Picoclaw: Ultra-efficient ai assistant in go. https://github.com/sipeed/picoclaw, 2026. GitHub repository. 4, 19

Fei Tang, Zhiqiong Lu, Boxuan Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Clawgui: A unified framework for training, evaluating, and deploying gui agents. *arXiv preprint arXiv:2604.11784*, 2026. 4, 19

ttguy0707. Cyberclaw – when ai starts black-box operations, you need x-ray vision. https://github.com/ttguy0707/CyberClaw, 2026. GitHub repository. 4, 19

Volcengine. Openviking: An open-source context database for ai agents. https://github.com/volcengine/OpenViking, 2025. Accessed: 2026-04-16. 4, 8

Daoyu Wang, Qingchuan Li, Mingyue Cheng, Jie Ouyang, Shuo Yu, Qi Liu, and Enhong Chen. Steppo: Step-aligned policy optimization for agentic reinforcement learning. *arXiv preprint arXiv:2604.18401*, 2026a. 4, 8, 9

Fiona Y Wang, Lee Marom, Subhadeep Pal, Rachel K Luu, Wei Lu, Jaime A Berkovich, and Markus J Buehler. Autonomous agents coordinating distributed discovery through emergent artifact exchange. *arXiv preprint arXiv:2603.14312*, 2026b. 4, 22, 23

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. 9

Jiayu Wang, Yifei Ming, Zixuan Ke, Shafiq Joty, Aws Albarghouthi, and Frederic Sala. Skillorchestra: Learning to route agents via skill transfer. *arXiv preprint arXiv:2602.19672*, 2026c. 4, 17

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, 2024. 3, 17

Linbo Wang, Yupeng Zheng, Qiang Chen, Shiwei Li, Yichen Zhang, Zebin Xing, Qichao Zhang, Xiang Li, Deheng Qian, Pengxuan Yang, et al. Latent-wam: Latent world action modeling for end-to-end autonomous driving. *arXiv preprint arXiv:2603.24581*, 2026d. 22

Xiangyu Wang, Donglin Yang, Wenhao Zheng, Zimu Tang, Zhibo Zhang, Yantong Zhong, Xiangyi Zheng, Qinan Liao, and Si Liu. Uav-claw: Embodied agent system for uav control based on openclaw, 2026e. URL https://prince687028.github.io/UAV-Claw/. Project webpage. Accessed: 2026-04. 4, 22

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*, 2026f. 4, 8, 9

Yuhang Wang, Feiming Xu, Zheng Lin, Guangyu He, Yuzhe Huang, Haichang Gao, Zhenxing Niu, Shiguo Lian, and Zhaoxiang Liu. From assistant to double agent: Formalizing and benchmarking attacks on openclaw for personalized local ai agent. *arXiv preprint arXiv:2602.08412*, 2026g. 3, 4, 11

Zijun Wang, Haoqin Tu, Letian Zhang, Hardy Chen, Juncheng Wu, Xiangyan Liu, Zhenlong Yuan, Tianyu Pang, Michael Qizhe Shieh, Fengze Liu, et al. Your agent, their asset: A real-world safety analysis of openclaw. *arXiv preprint arXiv:2604.04759*, 2026h. 11

Lukas Weidener, Marko Brkić, Phillip Lee, Martin Karlsson, Kevin Noessler, and Paul Kohlhaas. From agent-only social networks to autonomous scientific research: Lessons from openclaw and moltbook, and the architecture of clawdlab and beach. science. *arXiv preprint arXiv:2602.19810*, 2026a. 4, 22, 23

Lukas Weidener, Marko Brkić, Phillip Lee, Martin Karlsson, Kevin Noessler, and Paul Kohlhaas. From agent-only social networks to autonomous scientific research: Lessons from OpenClaw and Moltbook, and the architecture of ClawdLab and Beach.Science. *arXiv preprint arXiv:2602.19810*, 2026b. 4, 15

Mengsong Wu, Hao Hao, Shuzhen Bi, Keqian Li, Wentao Liu, Siyu Song, Hongbo Zhao, and Aimin Zhou. Scaling laws for educational ai agents. *arXiv preprint arXiv:2603.11709*, 2026. 4, 24

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. 5, 17

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025. 3, 17

Peng Xia, Jianwen Chen, Xinyu Yang, Haoqin Tu, Jiaqi Liu, Kaiwen Xiong, Siwei Han, Shi Qiu, Haonian Ji, Yuyin Zhou, et al. Metaclaw: Just talk–an agent that meta-learns and evolves in the wild. *arXiv preprint arXiv:2603.17187*, 2026. 4, 8, 9

Mingyang Xu, Junhao Chen, and Zaixi Zhang. Claw4science: A dataset and platform for the openclaw scientific agent ecosystem. *bioRxiv*, pages 2026–03, 2026. 23

Wenxian Yang, Hanzheng Qiu, Bangqun Zhang, Chengquan Li, Zhiyong Huang, Xiaobin Feng, Rongshan Yu, and Jiahong Dong. When openclaw meets hospital: Toward an agentic operating system for dynamic clinical workflows. *arXiv preprint arXiv:2603.11721*, 2026. 4, 23

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *Proc. ICLR*, 2022. 3, 5

Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, et al. Claw-eval: Toward trustworthy evaluation of autonomous agents. *arXiv preprint arXiv:2604.06132*, 2026. 4, 11

Zonghao Ying, Xiao Yang, Siyang Wu, Yumeng Song, Yang Qu, Hainan Li, Tianlin Li, Jiakai Wang, Aishan Liu, and Xianglong Liu. Uncovering security threats and architecting defenses in autonomous agents: A case study of openclaw. *arXiv preprint arXiv:2603.12644*, 2026. 3, 4, 10, 11, 12, 13, 24

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*, 2025a. 4, 8

Mingda Zhang. Scienceclaw, 2026. URL https://github.com/beita6969/ScienceClaw. GitHub repository. Accessed: 2026-04. 23

Yao Zhang, Chenyang Lin, Shijie Tang, Haokun Chen, Shijie Zhou, Yunpu Ma, and Volker Tresp. Swarmagentic: Towards fully automated agentic system generation via swarm intelligence. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1778–1818, 2025b. 4, 8, 10

Yuxuan Zhang, Yubo Wang, Yipeng Zhu, Penghui Du, Junwen Miao, Xuan Lu, Wendong Xu, Yunzhuo Hao, Songcheng Cai, Xiaochen Wang, et al. Clawbench: Can ai agents complete everyday online tasks? *arXiv preprint arXiv:2604.08523*, 2026. 4, 11

Yang Zhou, Xiaofeng Wang, Hao Shao, Letian Wang, Guosheng Zhao, Jiangnan Shao, Jiagang Zhu, Tingdong Yu, Zheng Zhu, Guan Huang, et al. Drivedreamer-policy: A geometry-grounded world-action model for unified generation and planning. *arXiv preprint arXiv:2604.01765*, 2026. 22

Ningyan Zhu, Huacan Wang, Jie Zhou, Feiyu Chen, Shuo Zhang, Ge Chen, Chen Liu, Jiarou Wu, Wangyi Chen, Xiaofeng Mou, et al. Semaclaw: A step towards general-purpose personal ai agents through harness engineering. *arXiv preprint arXiv:2604.11548*, 2026a. 4, 8

Xinyu Zhu, Yuzhu Cai, Zexi Liu, Cheng Wang, Fengyang Li, Wenkai Jin, Wanxu Liu, Zehao Bing, Bingyang Zheng, Jingyi Chai, et al. Evomaster: A foundational agent framework for building evolving autonomous scientific agents at scale. *arXiv preprint arXiv:2604.17406*, 2026b. 4, 8, 10

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024. 4, 8, 10

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. 21