A Lifecycle-Perspective Survey of Agent Skills in LLMs: From Construction to Ecosystems

Anonymous authors
Paper under double-blind review

Abstract

Skills—modular, reusable units of agent capability ranging from prompt templates to executable artifacts—have emerged as a central abstraction for LLMs to accumulate, compose, and refine experience. Yet research remains fragmented, with rapid progress scattered across construction, usage, training, safety, and evaluation. We present a unified, lifecycle-oriented survey of skill-based agent systems organized into six interconnected dimensions: *Construction and Evolution*, *Access and Composition*, *Model Integration via Training*, *Safety*, *Evaluation and Feedback*, and *Skill-Native Systems and Ecosystems*. A complementary taxonomy characterizes skills by representation, source, granularity, and role. For each dimension, we surface key challenges and forward-looking directions, including lifecycle management, compositional safety, dynamic utility evaluation, and ecosystem standardization. A continuously updated paper list is maintained at `https://anonymous.4open.science/r/awesome_agent_skills_anatomy-1C39`.

1 Introduction

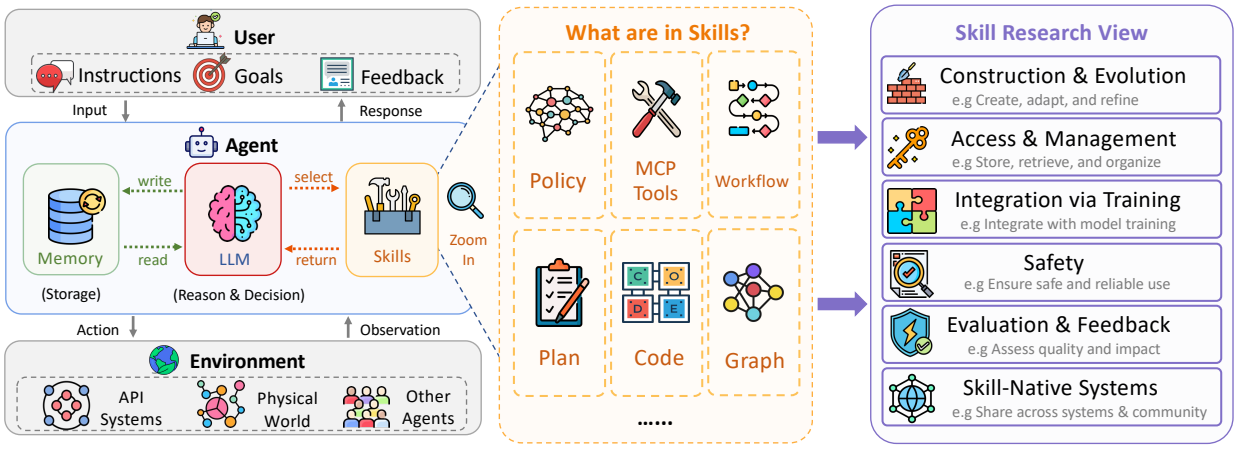


Figure 1: An overview of skill-based agent systems. **Left**: a typical agent architecture in which an LLM interacts with memory, skills, and the environment under user-provided instructions, goals, and feedback. **Middle**: skills can take diverse forms—including policies, MCP/tools, workflows, plans, code, and graphs—reflecting the breadth of the skill abstraction. **Right**: a unified research view organizes the skill landscape along six interconnected dimensions—construction, access, training-based integration, safety, evaluation, and skill-native systems—that together form a closed lifecycle loop.

Large language models (LLMs) have evolved from passive text predictors into agents (Xi et al., 2025; Luo et al., 2025) that act, reason, and interact with external environments. Whereas early progress was driven primarily by scaling pre-training data and model size (Kaplan et al., 2020; Hoffmann et al., 2022), recent

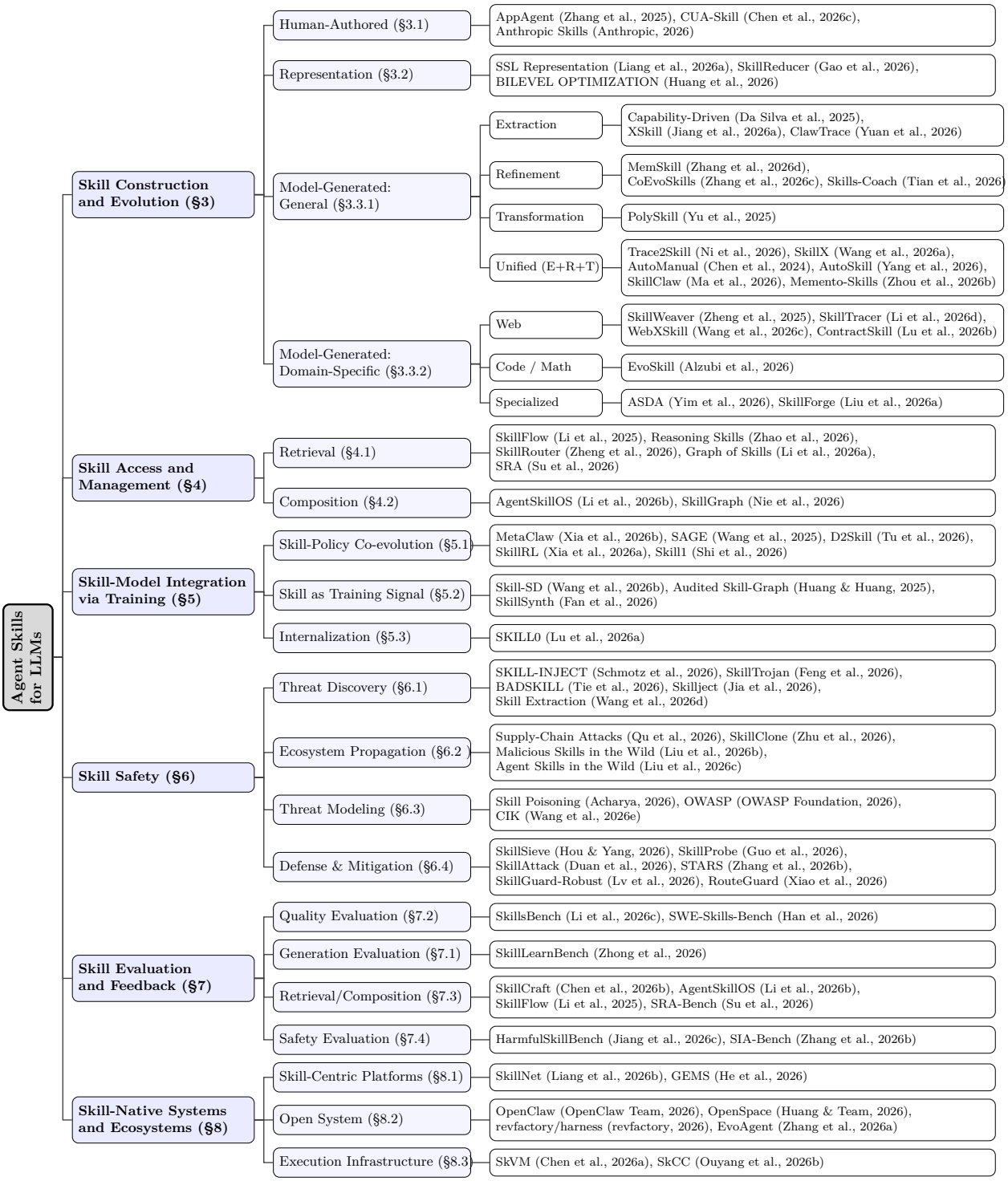


Figure 2: A taxonomy of agent skills for large language models.

advances increasingly rely on *post-training* (Ouyang et al., 2022) and *test-time* (Snell et al., 2024) mechanisms that elicit structured, reusable, and adaptive behaviors—shifting where agent capability lives, from raw model weights to the runtime interplay between models and their environments.

A central abstraction underlying this shift is the notion of **skills**. Rather than treating intelligence as a monolithic capability, recent work (Wang et al., 2023a) decomposes agent behaviors into modular, reusable units—ranging from prompt templates (Brown et al., 2020; Wei et al., 2022) and tool-use procedures (Schick et al., 2023) to learned sub-policies and structured artifacts such as `SKILL.md`.

Skills enable agents to move beyond one-shot generation toward *accumulating (Park et al., 2023), retrieving (Gao et al., 2023), composing (Khattab et al., 2023), and refining experience (Madaan et al., 2023)*, providing a foundation for scalable and continually evolving intelligence (Huang et al., 2023). Momentum has accelerated rapidly: in the past year, dozens of skill-centric frameworks have appeared in the literature (Zheng et al., 2025; Zhang et al., 2026d; Li et al., 2026a), while industry has begun to standardize skills as first-class artifacts through specifications such as Anthropic's `SKILL.md` (Anthropic, 2026) and emerging open skill marketplaces (OpenClaw Community, 2026).

Concurrent reviews and our scope. The rapid emergence of this area has motivated several recent reviews, each adopting a distinct lens. Xu & Yan (2026) organize the field around the `SKILL.md` specification and its integration with MCP, emphasizing architectural standardization but stopping short of training-based integration or lifecycle-level safety analysis. Jiang et al. (2026b) take a systems-of-knowledge view, distilling seven design patterns and an orthogonal representation–scope taxonomy with a notable security case study, yet treat retrieval and composition as a single stage and do not address skill internalization into model parameters. Zhou et al. (2026a) place skills within a broader "externalization" framework alongside memory, protocols, and harnesses, but their scope is organizational rather than lifecycle-oriented, leaving construction dynamics, training integration, and forward-looking open challenges largely unaddressed.

This survey. We offer a unified, lifecycle-oriented view of skill research in agent systems, organizing the field into six interconnected dimensions (Figure 1): (1) *Construction and Evolution*, (2) *Access and Composition*, (3) *Model Integration via Training*, (4) *Safety*, (5) *Evaluation and Feedback*, and (6) *Skill-Native Systems and Ecosystems*. Critically, these dimensions form a *closed loop* rather than a linear pipeline: skills constructed in (1) are retrieved and composed in (2), internalized into model parameters via (3), audited for safety in (4) and utility in (5), and deployed within ecosystems (6) that feed new experience back into (1) through reuse and evolution—tightening the loop with each iteration.

Our survey complements these works by adopting a *lifecycle-loop* lens that explicitly addresses the gaps above—covering training-based integration, distinguishing retrieval from composition, and foregrounding structural open challenges across all six dimensions. A complementary taxonomy (Figure 2) characterizes skills orthogonally along four dimensions: representation, source, granularity, and functional role.

Contributions.

- We present a comprehensive, lifecycle-oriented survey of skill-based LLM agent systems, covering recent works across six interconnected dimensions—with deeper coverage of training-based integration and retrieval–composition than concurrent surveys.
- We propose a unified taxonomy along representation, source, granularity, and functional role, enabling consistent comparison across heterogeneous methods.
- For each dimension, we identify structural challenges and forward-looking research directions, surfacing cross-cutting themes such as lifecycle management, compositional safety, dynamic utility evaluation, and ecosystem standardization.
- We maintain a curated, continuously updated paper list at `https://anonymous.4open.science/r/awesome_agent_skills_anatomy-1C39` to support the community.

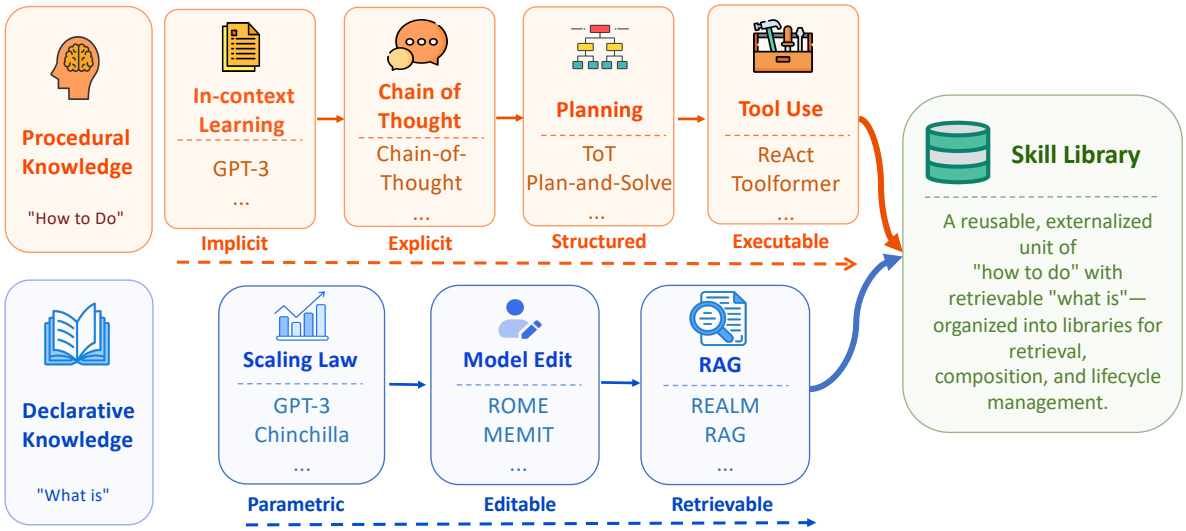


Figure 3: Evolution toward skills in agent systems. From in-context reasoning, agents externalize knowledge through RAG and capabilities through tool use; these two routes converge into reusable workflows, which are further abstracted into skills.

2 Skills in Agent Systems: From Emergence to Lifecycle

2.1 From Knowledge and Tools to Skills

Agents built on large language models (LLMs) operate over two distinct types of knowledge (Anderson, 1982): **declarative knowledge**—facts about the world ("what is"), and **procedural knowledge**—knowledge of how to act ("how to do"). The development of skills in LLM-based agents can be traced along two parallel lines, each addressing one type of knowledge.

Procedural line. Early LLM agents relied on **in-context learning (ICL)** (Brown et al., 2020) to perform new tasks from a few demonstrations, but the underlying procedural knowledge was implicit and ephemeral—it existed only within the context window and vanished after each interaction. **Chain-of-thought (CoT) prompting** (Wei et al., 2022) took a critical step by externalizing procedural knowledge into explicit natural-language reasoning chains, demonstrating that *how to solve* a task could be linguistically structured. **Planning methods** (Yao et al., 2023; Wang et al., 2023b) extended this further by organizing reasoning into structured multi-step procedures, transforming linear chains into trees or graphs of deliberation. **Tool use** (Schick et al., 2023; Yao et al., 2022) then grounded procedural knowledge in executable actions, enabling agents to interact with the environment—yet each execution remained one-shot, leaving no reusable trace. This limitation motivated the notion of **skills** (Wang et al., 2023a): structured, reusable units of procedural knowledge that can be extracted from experience and invoked across tasks.

Declarative line. In parallel, the research community pursued a separate question: how to equip LLMs with sufficient declarative knowledge. Early efforts focused on **scaling** (Brown et al., 2020; Hoffmann et al., 2022), compressing ever-larger corpora into model weights—but parametric knowledge remained static and difficult to update. **Model editing** (Meng et al., 2022a;b) addressed this by surgically modifying specific parameters to update or correct factual knowledge without full retraining. **Retrieval-augmented generation (RAG)** (Lewis et al., 2020) took a more fundamental step by externalizing declarative knowledge into dynamic repositories, enabling on-demand retrieval at inference time. As skills grew more sophisticated, they naturally incorporated RAG-style retrieval as an information access layer, drawing on external declarative knowledge during execution (Wang et al., 2023a).

Table 1: Comparison of skills with related concepts. Skills capture reusable procedural knowledge that can access information via memory (*) and trigger actions via tools (†), subsumming the capabilities of tools, memory, and plans.

Concept	Reusable	Procedural	Informational	Executable
Tool	✗	✗	✗	✓
Memory	✓	✗	✓	✗
Plan	✗	✓	✗	✗
Skill	✓	✓	✓*	✓†

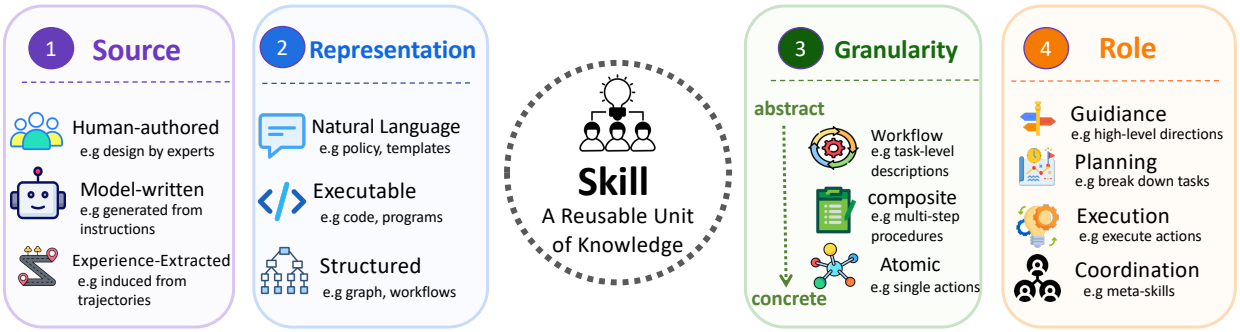


Figure 4: Four design dimensions characterizing skills as a unified abstraction. **Source** captures how skills are produced (human-authored, model-written, or extracted from experience); **Representation** describes how they are encoded (natural language, executable code, or structured graphs); **Granularity** ranges from atomic actions to composite procedures and abstract workflows; and **Role** reflects their functional purpose in agent systems (guidance, planning, execution, or coordination). These dimensions are largely orthogonal, and a single skill can be positioned along all four.

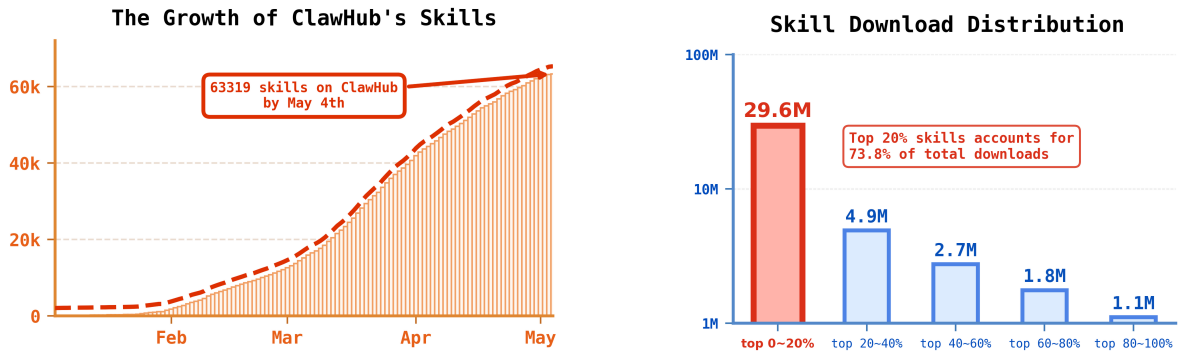
Convergence: skill library. The two lines converge at the **skill library**: the procedural line contributes reusable *how-to* units, while the declarative line contributes the retrieval mechanism to manage and access them at scale. A skill library does for procedural knowledge what RAG did for declarative knowledge—explicit storage, dynamic retrieval, and on-demand invocation (Wang et al., 2023a; Qian et al., 2023). As libraries grow larger, efficiently selecting the right skill under noisy or ambiguous conditions emerges as a central challenge, which we address in this survey.

2.2 What is a Skill? Definition, Scope, and Design Dimensions

We define a **skill** as a reusable procedural unit that enables an agent to solve a class of tasks by encoding *how* to act—that is, a structured decision procedure that can be reused, adapted, and composed across contexts.

Unlike static knowledge representations, a skill captures *actionable* knowledge: it specifies not only what information is relevant, but also how to transform observations into decisions and actions over potentially multiple steps. As such, skills naturally support compositionality and reuse, allowing agents to generalize beyond individual task instances.

Distinction from related concepts. A skill is closely related to, but fundamentally different from, several common abstractions in agent systems. **Tools** expose executable functionalities but contain no procedural knowledge—they do not specify when, why, or how to be invoked in context. **Memory** stores facts, past interactions, or retrieved knowledge, but does not prescribe decision-making procedures. **Plans** specify task-specific action sequences but are typically ephemeral and non-reusable across tasks. In contrast, a skill encapsulates reusable procedural knowledge that can leverage tools and memory, while remaining transferable across tasks and contexts (Table 1).



(a) Cumulative skill growth.

(b) Download distribution by quintile.

Figure 5: Empirical evidence from ClawHub illustrating that rapid skill creation coexists with a heavy-tailed usage distribution. **(a)** The registry grew from near-zero to over 63,000 skills between January and May 2026. **(b)** On a log scale, the top 20% of skills account for 73.8% of total downloads (∼29.6M), while the bottom 80% combined receive only 26.2% (∼10.5M).

Design dimensions. Skills can be systematically characterized along four orthogonal design dimensions:

(1) **Source** specifies *how a skill is produced*: human-authored by experts, generated by models from natural-language instructions, or extracted from interaction trajectories and execution histories.

(2) **Representation** specifies *the form in which a skill is encoded*: natural-language formats (e.g., prompts or templates), executable representations (e.g., code or programs), or structured abstractions (e.g., graphs or workflows).

(3) **Granularity** specifies *the level of abstraction at which a skill operates*, ranging from atomic actions (e.g., a single API call), to composite procedures (e.g., multi-step routines), to task-level workflows (e.g., end-to-end task templates).

(4) **Role** specifies *how a skill functions within an agent*: providing high-level guidance, supporting planning and decomposition, executing primitive actions, or—as *meta-skills*— coordinating the selection and composition of other skills (Section 4.2).

Figure 4 summarizes these dimensions, which together provide the conceptual scaffolding for the taxonomy and discussions throughout this survey.

2.3 The Growth and Quality Gap of Skills

To motivate why a structured study of skills is timely, we examine the state of a representative open skill registry, ClawHub (OpenClaw Community, 2026), an emerging community-driven platform for publishing, versioning, and discovering text-based agent skills.

Skill creation is accelerating rapidly. Figure 5a shows that ClawHub grew from a nascent registry to over 63,000 published skills in roughly four months (January–May 2026), with growth accelerating sharply from March onward. This explosive supply mirrors the rise of community-driven artifact ecosystems in adjacent domains—models on Hugging Face, datasets on Kaggle, packages on PyPI—and signals that skills are emerging as a first-class artifact type that practitioners actively author, share, and reuse.

Yet quality and usage are highly concentrated. However, sheer volume does not imply uniform utility. Figure 5b reveals a heavy-tailed download distribution: the top 20% of skills account for **73.8%** of all downloads (∼29.6M), while the bottom 80% combined contribute only ∼10.5M. Within the long tail, the

lowest quintile averages fewer than 100 downloads per skill, suggesting that the vast majority of published skills see little real-world use.

Implications. This pattern—*rapid creation, concentrated utility*—is precisely the regime in which structured research is most needed. It surfaces fundamental questions across the full skill lifecycle: how should skills be *constructed* to be genuinely reusable rather than redundant (§3)? How can agents reliably *retrieve and compose* the few high-utility skills from a long tail of low-quality ones (§4)? When and how should skills be *absorbed into model parameters* versus kept externally manageable (§5)? How can the ecosystem be made *safe* when malicious or low-quality skills circulate at scale (§6)? How should skill libraries be *evaluated* for actual contribution to agent performance, not merely for existence (§7)? And how can *ecosystem-level systems* support quality assurance, deduplication, and lifecycle management (§8)? The empirical reality of today's skill registries makes these questions practical, not hypothetical— motivating the unified, lifecycle-oriented view we develop next.

2.4 A Unified View of Skill Research

We organize the skill research landscape into six interconnected dimensions capturing the lifecycle of skills in agent systems.

- **Skill Construction and Evolution:** creation and refinement of skills, from human-authored designs to model-generated ones distilled from experience.
- **Skill Access and Management:** retrieval, and composition of skills to meet task demands.
- **Skill–Model Integration via Training:** how training bridges skills and model capabilities, including co-evolution and internalization.
- **Skill Safety:** ensuring trustworthiness, including harmful content, unsafe execution, and auditability.
- **Skill Evaluation and Feedback:** assessing skills across generation, usage, and retrieval, and using feedback for improvement.
- **Skill-Native Systems and Ecosystems:** system-level organization of skills, including skill-centric design and portable execution.

These dimensions form a closed-loop lifecycle: skills are constructed, accessed, and continuously reshaped through evaluation, feedback, and safety constraints—serving as the organizing principle for this survey.

3 Skill Construction and Evolution

Skills are not static artifacts; they are continuously constructed, adapted, and refined. We organize this space along three axes: **human-authored skills**, which are manually designed; **model-generated skills**, which are automatically synthesized to scale; and **skill representation**, which determines how skills are structured and expressed for machine execution.

3.1 Human-Authored Skills

The earliest paradigm equips agents with skills through direct human authorship. Early efforts focus on tool- and API-level abstractions, where systems such as Toolformer (Schick et al., 2023), ToolBench (Guo et al., 2024), RestGPT (Song et al., 2023), and Gorilla (Patil et al., 2024) curate API descriptions to ground language models in external services.

At a higher abstraction level, AppAgent (Zhang et al., 2025) and CUA-Skill (Chen et al., 2026c) hand-design GUI and computer-use procedures as modular skills, while Anthropic's `SKILL.md` (Anthropic, 2026) standardizes skills as auditable, version-controlled artifacts.

Summary. Human-authored skills offer high reliability and auditability through expert-crafted artifacts—from API-level descriptions to standardized specifications like `SKILL.md`—but scale poorly due to the manual effort required for authoring and maintenance.

3.2 Skill Representation

Beyond skill construction, an emerging line of work focuses on how skills are *represented* as first-class objects—shifting from unstructured text to structured, optimized, and machine-interpretable forms.

From text to structure. Early approaches treat skills as free-form text, which limits composability and reliable execution. SSL (Liang et al., 2026a) addresses this by mapping textual skill descriptions into a three-layer graph (*Scheduling*, *Structural*, *Logical*), enabling explicit control over when skills are invoked, how they are organized, and what logic they encode.

From structure to efficiency. As skill libraries scale, representation cost becomes a bottleneck. SkillReducer (Gao et al., 2026) compresses and rewrites skill representations to reduce token usage while preserving functional equivalence, highlighting the importance of efficiency in large-scale skill deployment.

From representation to optimization. Beyond static design, recent work treats skill representation itself as an optimization problem. Huang et al. (2026) formulate structure search as a bilevel process, combining MCTS for exploring component organization with LLM-based refinement for content, enabling automated discovery of effective skill structures.

Overall, these works illustrate a clear progression: from *structuring* skills for interpretability, to *compressing* them for efficiency, and ultimately to *optimizing* their representations for performance—suggesting that representation is not merely a design choice, but a key axis for scaling skill-based systems.

3.3 Model-Generated Skills

To overcome the scalability bottleneck of human authorship, a second paradigm enables language models to autonomously generate and evolve skills. As illustrated in Figure 6, model-generated skill construction can be unified into three fundamental operations:

- **Extraction** — creating skills from scratch, e.g., summarizing a trajectory into a new procedure.
- **Refinement** — optimizing existing skills via execution feedback or verification.
- **Transformation** — restructuring skills into more modular or abstract forms for reuse.

We organize the literature along two axes: Section 3.3.1 reviews *general* skill generation grouped by dominant operation; Section 3.3.2 reviews *domain-specific* generation under particular environments such as the Web or enterprise settings.

3.3.1 General Skill Generation

General-purpose skill generation can be understood as a progressive pipeline that moves from *extracting* reusable knowledge from experience, to *refining* it through feedback, and finally to *transforming* it for generalization.

From experience to skills (Extraction). The first step is to lift raw interaction trajectories into reusable skill abstractions. Capability-Driven Skill Generation (Da Silva et al., 2025) synthesizes skills from capability descriptions via retrieval-augmented generation, while XSkill (Jiang et al., 2026a) distills multimodal rollouts into structured skills using a dual-stream architecture that separates action-level experience from task-level abstraction. ClawTrace (Yuan et al., 2026) further improves this stage by introducing fine-grained, cost-aware trace filtering to retain only high-value trajectories for skill induction.

From skills to better skills (Refinement). Once constructed, skills must be continuously refined through feedback, failure analysis, and iterative optimization. A central theme in this line of work is building closed-loop refinement systems that autonomously diagnose weaknesses and improve skill quality over time.

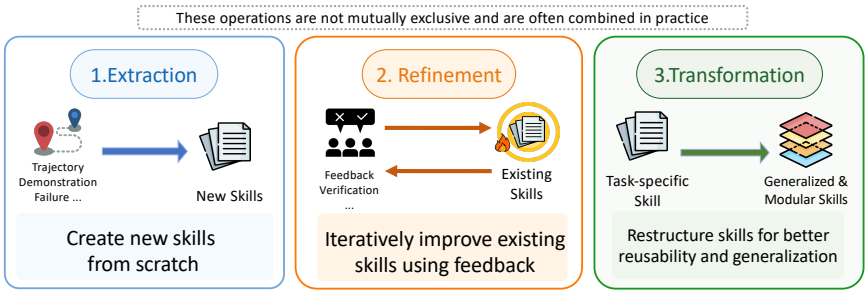


Figure 6: Three core operations in skill construction. **Extraction** creates skills from raw experience or demonstrations. **Refinement** improves existing skills via feedback loops. **Transformation** restructures task-specific skills into generalized, modular forms for broader reuse. These operations are often combined in practice.

MemSkill (Zhang et al., 2026d) reframes memory operations as evolvable skills, introducing a controller–executor–designer framework that jointly performs skill retrieval, memory extraction, and failure-driven refinement. CoEvoSkills (Zhang et al., 2026c) further removes dependence on ground-truth supervision by co-evolving a Skill Generator with a Surrogate Verifier, enabling autonomous refinement of complex multi-file skill packages through iterative feedback. Similarly, Skills-Coach (Tian et al., 2026) studies training-free skill optimization through automated evaluation and refinement, leveraging benchmark-driven feedback to substantially improve skill reliability across different difficulty levels.

From specific to general (Transformation). Beyond correctness, skills must generalize across contexts. PolySkill (Yu et al., 2025) introduces polymorphic abstraction that disentangles a skill's goal (*what*) from its implementation (*how*), allowing skills to be re-instantiated across environments (e.g., different websites) and improved through curriculum-based reuse.

Closing the loop (Unified E+R+T). Recent approaches integrate extraction, refinement, and transformation into closed-loop systems, differing primarily in their optimization objectives.

- **Trajectory distillation.** Methods such as Trace2Skill (Ni et al., 2026) distill trajectory-level experiences into transferable skills, while SkillX (Wang et al., 2026a) further organizes them into hierarchical structures (strategic → functional → atomic) for modular reuse.
- **Lifelong personalization.** AutoSkill (Yang et al., 2026) continuously extracts and refines user-specific skills from interactions, and SkillClaw (Ma et al., 2026) extends this paradigm to shared, multi-user settings.
- **Agent-designing-agent.** Memento-Skills (Zhou et al., 2026b) treats the skill library as a first-class object, enabling iterative agent redesign through reflective read–write cycles over structured repositories.
- **Environment-rule externalization.** AutoManual (Chen et al., 2024) externalizes environment dynamics into human-readable manuals, facilitating cross-model transfer and reducing reliance on implicit knowledge.

3.3.2 Domain-Specific Skill Generation

While general methods aim for broad applicability, a parallel line of work instantiates the same extraction–refinement–transformation (E/R/T) pipeline under domain-specific constraints, where environment structure, supervision signals, and execution semantics reshape how skills are generated and improved.

Web agents. Web environments provide structured interaction traces (e.g., DOM trees, URLs), enabling fine-grained skill extraction and localized refinement. SkillWeaver (Zheng et al., 2025) discovers website-specific skills through self-practice and distills them into reusable APIs, while WebXSkill (Wang et al., 2026c) pushes extraction to a finer granularity by mining parameterized skills from synthetic trajectories organized as a URL-indexed graph.

Table 2: Model-generated skill construction methods grouped by their target scope. E: Extraction; R: Refinement; T: Transformation. Domain indicates whether the method targets general agent tasks or a specific application domain.

Work	E	R	T	Domain	Key Idea
General Skill Generation					
Capability-Driven	✓			General	Generates skills via RAG over libraries from capability specs
Trace2Skill	✓	✓	✓	General	Distills trajectory-local lessons into transferable skills
AutoManual	✓	✓		General	Planner-Builder dual agents extract and refine rules online
SkillX	✓	✓	✓	General	Builds three-tier skill knowledge bases with iterative revision
PolySkill		✓	✓	General	Decouples goals and implementations via polymorphic abstraction
MemSkill	✓	✓		General	Treats memory operations as evolvable skills via closed-loop learning
CoEvoSkills		✓		General	Co-evolves skill generator and surrogate verifier
Skills-Coach		✓		General	Benchmark-driven training-free optimization of real-world skills
AutoSkill	✓	✓	✓	General	Lifelong learning of personalized skills via self-evolution
SkillClaw	✓	✓	✓	General	Aggregates multi-user experience for collective skill evolution
Memento-Skills	✓	✓	✓	General	Read-Write reflective learning for memory-based skill design
XSkill	✓	✓		General	Dual-stream distillation of action-level experiences and task-level skills
ClawTrace	✓	✓		General	Cost-aware tracing to retain high-value trajectories for skill distillation
Domain-Specific Skill Generation					
SkillWeaver	✓	✓	✓	Web	Discovers and hones reusable APIs transferable across web agents
WebXSkill	✓			Web	Mines parameterized action skills with step-level guidance
SkillTracer	✓	✓	✓	Web	Plan-graph failure attribution with targeted structural repair
ContractSkill		✓		Web	Contract-based skills with deterministic verification and local repair
EvoSkill	✓	✓	✓	Code	Iteratively evolves coding skills via failure analysis
ASDA	✓	✓		Finance	Teacher-driven failure clustering for financial reasoning skills
SkillForge	✓	✓		Cloud	Failure-driven refinement of cloud-support skills from tickets

On the refinement side, SkillTracer (Li et al., 2026d) performs targeted repair over attributed plan graphs, and ContractSkill (Lu et al., 2026b) enforces correctness by formulating skills as executable contracts, enabling deterministic verification and local edits. Together, these methods emphasize *structure-aware extraction* and *verifiable refinement* enabled by the web setting.

Code and math. In programmatic domains, skills emerge from iterative failure and correction with strong executability signals. EvoSkill (Alzubi et al., 2026) exemplifies this paradigm by continuously extracting reusable skills from failed coding trajectories and refining them through execution feedback, forming a tight improvement loop grounded in testable correctness.

Specialized enterprise domains. In domain-intensive settings, skill generation is tightly coupled with external knowledge and workflows. SkillForge (Liu et al., 2026a) grounds skills in domain knowledge bases and historical tickets, refining them through failure-driven updates, while ASDA (Yim et al., 2026) focuses on financial reasoning by clustering failure modes and synthesizing structured skill artifacts (e.g., workflows and exemplars) for dynamic reuse. These approaches highlight the role of *domain priors* and *knowledge grounding* in guiding both extraction and refinement.

Overall, domain-specific methods do not change the core E/R/T paradigm, but *specialize it*: web agents leverage structural signals for precise extraction and verification, code domains rely on executable feedback for tight refinement loops, and enterprise settings incorporate domain knowledge to guide skill formation and reuse.

3.4 Summary

This section traced skill construction and evolution along three complementary threads—**human-authored skills** (§3.1), **skill representation** (§3.2), and **model-generated skills** (§3.3). Across these threads, a clear trend emerges: skill construction is shifting from *static authorship* toward *dynamic, closed-loop evolution*, where skills are not only generated but continuously verified, restructured, and reused without explicit parameter updates.

Despite this progress, three fundamental challenges remain:

- **Generalization vs. specialization.** Skills frequently overfit to the trajectories or environments from which they are derived, and current transformation techniques (e.g., polymorphic abstraction) are still limited to narrow transfer scenarios.
- **Quality assurance without ground truth.** Most refinement loops rely on self- or surrogate verification (e.g., CoEvoSkills, MemSkill), which lacks correctness guarantees and risks compounding errors over iterations.
- **Lifecycle and evaluation gaps.** As skill libraries grow, there is little support for deduplication, versioning, and principled deprecation, and no standardized benchmark exists to measure skill quality, reuse, or long-term impact—a gap that motivates our discussion of skill evaluation in §7.

Open Challenges

- **Skill Library Consistency.** How to maintain conflict-free, non-redundant skill sets as libraries grow with multi-source contributions remains an open problem—going beyond the lifecycle gap noted above toward principled merging and arbitration.
- **Lifecycle-Adaptive Representation.** Skill representations should adapt across their lifecycle (richer during refinement, compact at deployment) while supporting correct and efficient updates under evolving environment rules.

4 Skill Access and Composition

After skills are constructed, agents must determine *which* skills to use and *how* to combine them. We view this process as a two-stage pipeline: **skill retrieval**, which selects relevant skills from a library, and **skill composition**, which organizes them into executable structures.

4.1 Skill Retrieval

Skill retrieval maps a task context to a subset of relevant skills under constraints of quality, coverage, and efficiency. A key challenge is balancing *selectivity* and *completeness*: exposing too many skills introduces noise, while overly selective retrieval risks missing critical components.

Existing approaches differ in how they represent and access skill libraries. SkillFlow (Li et al., 2025) identifies quality and coverage as fundamental bottlenecks in large-scale skill repositories. SkillRouter (Zheng et al., 2026) shows that routing decisions depend critically on access to full skill content rather than compressed representations. SRA (Su et al., 2026) formalizes a *retrieve–incorporate–apply* paradigm, demonstrating that selective exposure can outperform full skill injection.

Beyond independent retrieval, recent work models structural dependencies among skills. Graph of Skills (Li et al., 2026a) constructs an offline dependency graph and retrieves skill bundles via Personalized PageRank, enabling dependency-aware selection. AgentSkillOS (Li et al., 2026b) further integrates retrieval with execution, while Thinking with Reasoning Skills (Zhao et al., 2026) extends retrieval mechanisms to reasoning-centric tasks.

4.2 Skill Composition and Orchestration

Given retrieved skills, the next challenge is to compose them into coherent execution strategies. This requires not only selecting compatible skills but also organizing them into structured workflows.

Existing approaches highlight different composition paradigms. AgentSkillOS (Li et al., 2026b) organizes skills into DAG-based capability trees, showing that structured composition can improve performance beyond individual skill quality. SkillGraph (Nie et al., 2026) models skills as nodes in a graph-based multi-agent system, enabling decentralized coordination. SkillMOO (Gong et al., 2026) formulates composition as a multi-objective optimization problem, where pruning and substitution outperform naive accumulation.

As skill libraries scale, the bottleneck increasingly shifts from retrieval to effective coordination, making composition and orchestration central to scalable skill-based agents.

4.3 Summary

This section traced skill access and composition along two stages—**skill retrieval** (§4.1), which selects relevant skills under quality and coverage constraints, and **skill composition** (§4.2), which organizes them into executable structures. Across both stages, the field has evolved from isolated, content-based mechanisms toward structured pipelines that incorporate dependency-aware retrieval and optimization-based composition for large-scale skill libraries.

Despite this progress, three fundamental challenges remain:

- **Static structures in dynamic environments.** Most retrieval and composition methods rely on offline-constructed graphs, fixed workflows, or static skill representations, limiting adaptability to open-ended or partially observed task contexts.
- **Implicit assumption of skill compatibility.** Existing methods assume retrieved skills can be freely combined, with little support for detecting conflicts, redundancies, or interaction effects among co-selected skills.
- **Decoupled retrieval and composition.** Selection and coordination are typically treated as independent stages without unified optimization, and neither is jointly tuned with the underlying skill descriptions on which they depend.

Open Challenges

- **Conflict-Aware Selection.** Beyond identifying conflicts as a gap, future work should develop principled mechanisms. e.g., compatibility checks, arbitration policies—to resolve them at selection or composition time.
- **Efficiency-Aware Composition.** Skills vary substantially in execution cost and quality; jointly optimizing skill selection and composition under efficiency constraints (latency, tokens, success rate) remains an open direction.
- **Co-designing Skill Descriptions with Retrieval.** Since skill descriptions vary across human authors, models, and generation methods, retrieval systems and description conventions should be co-designed under a unified style to ensure reliable matching.

5 Skill-Model Integration via Training

This section studies how models *update their parameters* using skill-related signals, transitioning from external skill usage to internalized capability. We organize existing approaches as a progression from **co-evolving with skills**, to **learning from skills as supervision**, and finally to **internalizing skills into model parameters**.

5.1 Skill–Policy Co-evolution

The first paradigm couples skill evolution with policy optimization, allowing agents to continuously improve external skill libraries and internal decision policies during training. A central theme across this line of work is the formation of a closed feedback loop: stronger policies generate better trajectories for skill extraction and refinement, while improved skills provide higher-quality guidance for subsequent policy learning.

Existing approaches mainly differ in how skills are organized and integrated into reinforcement learning. Some methods focus on recursive co-evolution between policies and hierarchical skill memories, such as SkillRL (Xia et al., 2026a), which maintains structured *SkillBanks* for adaptive retrieval and continual refinement. Others integrate skills directly into the RL optimization process itself. For example, SAGE (Wang et al., 2025) introduces skill-aware rewards and sequential rollouts to jointly optimize skill generation and usage, while

Table 3: Training paradigms for skill–model integration. Methods mainly differ in whether skills co-evolve with policies, serve as training signals, or are internalized into model parameters.

Method	**Paradigm**	**Core Idea**
Skill–Policy Co-evolution		
MetaClaw	Alternating optimization	Iteratively improves skills and policy together.
SkillRL	Recursive co-evolution	Co-evolves hierarchical SkillBanks and RL policies.
SAGE	Skill-integrated RL	Integrates skill generation and usage into RL.
D2Skill	Utility-aware co-training	Jointly optimizes dual-granularity skills and policy.
Skill1	Unified RL optimization	Unifies skill selection, usage, and distillation.
SkillOS	Skill curation learning	Learns insertion, update, and deletion of skills.
Skill as Training Signal		
Skill-SD	Skill-conditioned distillation	Uses skills as privileged teacher guidance.
Audited Skill Graphs	Verifiable supervision	Uses auditable skill graphs for safe optimization.
SkillSynth	Skill-graph synthesis	Generates training tasks from skill graphs.
Skill Internalization		
SKILL0	Curriculum internalization	Gradually removes external skills during training.

D2Skill (Tu et al., 2026) separates task-level and step-level skills to support both high-level planning and fine-grained correction.

Another direction studies unified lifecycle optimization, where skill selection, utilization, distillation, and even repository management are learned together within the training loop. Skill1 (Shi et al., 2026) optimizes the entire skill lifecycle using only task-level rewards, while SkillOS (Ouyang et al., 2026a) further treats skill curation itself as a learnable policy, training a dedicated curator model to perform insertion, updating, and deletion operations over evolving skill repositories.

Meanwhile, MetaClaw (Xia et al., 2026b) combines training-free skill evolution with periodic RL-based policy updates, bridging online skill adaptation and offline parameter optimization.

Overall, these methods blur the boundary between skills and policies, transforming skills from static external tools into dynamically evolving components of the learning process.

5.2 Skill as Training Signal

A second line of work uses skills as structured training signals to improve model capabilities. Instead of executing skills directly at inference time, these methods treat skills as supervision, reward signals, or structured sources for training data generation.

One direction uses skills as privileged guidance during optimization. Skill-SD (Wang et al., 2026b) summarizes successful trajectories into natural-language skills and uses them as teacher-only information in a self-distillation framework, allowing the student policy to gradually internalize skill guidance through RL and reverse-KL training.

Another direction leverages structured skill representations to improve training reliability and safety. Audited Skill Graphs (Huang & Huang, 2025) replace opaque parameter updates with verifiable skill-level supervision and reward signals, making self-improvement more transparent and controllable.

Skills can also serve as compositional primitives for data synthesis. SkillSynth (Fan et al., 2026) constructs training trajectories by sampling over skill graphs, generating diverse terminal tasks for supervised fine-tuning.

Compared with skill–policy co-evolution, this paradigm decouples skill construction from policy optimization, leading to more stable and interpretable training pipelines, though often with reduced adaptability during online learning.

5.3 Skill Internalization

The final stage asks whether external skills can be absorbed into model parameters, eliminating the need for explicit skill usage at inference time. SKILL0 (Lu et al., 2026a) gradually removes skill context during training, forcing the model to internalize skill-guided behaviors. Similarly, Skill-SD (Wang et al., 2026b) supports internalization through distillation, enabling skill-free deployment after training. These approaches represent a shift from *using* skills to *becoming* the skills, trading flexibility for efficiency and simplicity at inference.

5.4 Summary

This section traced skill–model integration along three paradigms—**skill– policy co-evolution** (§5.1), **skill as training signal** (§5.2), and **skill internalization** (§5.3). Across these paradigms, the field has shifted from *using* external skills to *absorbing* them into model parameters, trading adaptability for inference efficiency at progressively deeper levels of integration.

Despite this progress, three fundamental challenges remain:

- **Paradigm-specific trade-offs without unification.** Co-evolution offers adaptability but suffers from training instability and scalability issues; supervision-based methods simplify optimization but depend on imperfect skill signals; internalization improves efficiency but struggles with compositional or evolving skills. No principled framework jointly optimizes across these paradigms.
- **Supervision quality bottleneck.** When skills serve as training signals or distillation targets, their correctness and coverage directly bound the learned policy—yet most methods rely on self-generated or weakly verified skills, propagating errors into model parameters.
- **Incomplete internalization of compositional skills.** Current internalization techniques often fail to absorb multi-step or interacting skills, leaving persistent gaps between skill-assisted and skill-free inference.

Open Challenges

- **Skill Quality Filtering in Training.** As skills are dynamically generated and evolved, training pipelines should incorporate principled filtering and arbitration mechanisms to prevent low-quality or conflicting skills from corrupting parameter updates.
- **Unified Training Framework Across Paradigms.** Co-evolution, supervision, and internalization are currently studied in isolation; future work should explore staged or hybrid pipelines that combine their strengths. e.g., co-evolving early, distilling mid-training, and internalizing at deployment.

6 Skill Safety

As agents construct, share, and reuse skills, safety risks emerge at the level of *skills themselves*. We organize this space as a threat lifecycle: from identifying skills as an attack surface, to understanding how risks propagate across ecosystems, to developing systematic threat models, and finally to designing defenses. This progression reflects a shift from empirical observation of attacks to conceptual abstraction of their underlying structure.

6.1 Threat Discovery

Early work establishes skills as a distinct attack surface, revealing agents' over-trust in external skill artifacts. SKILL-INJECT (Schmotz et al., 2026) demonstrates injection attacks via malicious skill files, while Skill-Trojan (Feng et al., 2026) and BADSKILL (Tie et al., 2026) extend this to backdoor-style threats, including *model-in-skill* poisoning that bypasses code-level inspection. Skillject (Jia et al., 2026) further automates these attacks via trace-driven closed-loop optimization.

Beyond injection, attacks expand along two additional directions. First, proprietary skills can be extracted through black-box interaction (Wang et al., 2026d), exposing intellectual property and internal logic. Second,

runtime threats emerge: skills may appear benign at install time but fetch malicious instructions after deployment, and even security scanners can be weaponized to execute hidden payloads while reporting clean results (Tal, 2026). These findings collectively establish skills as a rich and underexplored attack surface.

6.2 Ecosystem Propagation

Following initial threat discovery, subsequent work shows that risks do not remain isolated but propagate through shared skill ecosystems. Supply-chain attacks spread malicious skills via reuse (Qu et al., 2026), and SkillClone (Zhu et al., 2026) demonstrates how cloning amplifies both errors and embedded backdoors.

Large-scale empirical studies confirm that such risks are widespread in practice (Liu et al., 2026b;c), uncovering prompt injection, over-privileged execution, and data leakage. Industry measurements further quantify this risk: among 3,984 real-world skills, 36.82% contain at least one vulnerability, with 13.4% rated critical (Snyk, 2026). These results highlight that safety issues are systemic rather than incidental, emerging from ecosystem-level reuse and distribution. At this stage, the focus is on *how risks spread* and how severe they become in real-world deployments.

6.3 Threat Modeling

As risks propagate across ecosystems, understanding them requires moving beyond empirical observations toward systematic threat modeling. Rather than asking how attacks spread, this line of work seeks to formally characterize *what attacks are* and how they can be structured and categorized.

Skill Poisoning (Acharya, 2026) integrates injection, supply-chain, and compositional risks into a unified taxonomy with a multi-layer defense architecture. Complementing this, Wang et al. (2026e) propose the CIK taxonomy (Capability, Identity, Knowledge), showing that poisoning along any single dimension can significantly increase attack success rates.

OWASP AST10 (OWASP Foundation, 2026) further distills field-confirmed risks into a standardized taxonomy and introduces a Universal Skill Format for cross-platform mitigation. Together, these efforts shift the field from observing how attacks manifest in practice to building structured abstractions that enable systematic reasoning about skill-level security.

6.4 Defense and Mitigation

Building on these models, defense mechanisms are developed across multiple stages of the skill lifecycle.

Pre-deployment filtering. SkillSieve (Hou & Yang, 2026) performs hierarchical triage for skill vetting, while SkillGuard-Robust (Lv et al., 2026) provides role-aware auditing robust to semantics-preserving rewrites. Cisco Skill Scanner (Cisco AI Defense, 2026) extends coverage with multi-layer analysis, including LLM-based semantic inspection integrated into CI/CD pipelines.

Runtime protection. Defenses increasingly rely on context- and model-aware signals. STARS (Zhang et al., 2026b) introduces request-conditioned risk scoring, and RouteGuard (Xiao et al., 2026) detects poisoning via internal signals such as attention hijacking and hidden-state shifts, recovering attacks missed by lexical screening.

Proactive evaluation. SkillProbe (Guo et al., 2026) enables multi-agent auditing, while SkillAttack (Duan et al., 2026) applies adaptive red teaming across diverse harm categories.

Despite these advances, defenses remain fragmented, with limited coverage of runtime and compositional threats, leaving a gap between threat complexity and protection capability.

6.5 Summary

This section traced skill safety along a threat lifecycle—from **threat discovery** (§6.1), through **ecosystem propagation** (§6.2) and **threat modeling** (§6.3), to **defense and mitigation** (§6.4). Across these efforts,

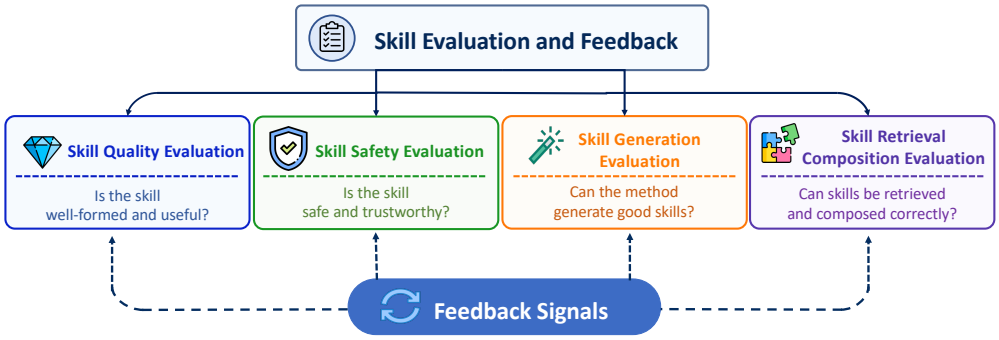


Figure 7: Overview of skill evaluation and feedback. Evaluation spans generation, intrinsic quality, usage (retrieval & composition), and safety, forming a feedback loop for improving skill-based systems.

a consistent pattern emerges: the field has moved from empirical observation toward systematic abstraction, yet defenses continue to lag behind the threats they target.

Attacks have expanded from isolated injection to ecosystem-level propagation and compositional exploitation, while defenses remain largely confined to single-stage, static analysis. This asymmetry points to three structural gaps.

- **Compositional blind spot.** Individually safe skills may produce dangerous behaviors when combined, yet no existing defense operates at the composition level—the most complex attack surface remains the least protected.
- **Lifecycle fragmentation.** Pre-deployment filters and runtime monitors are developed in isolation, leaving transitions between stages (e.g., install-time-clean but post-deployment-malicious skills) as a persistent blind spot.
- **Static defenses against adaptive adversaries.** Most defenses rely on heuristics or lexical inspection and are readily bypassed by semantics-preserving rewrites or context-aware attacks. While RouteGuard's internal-signal approach begins to close this gap, fully adaptive defenses remain an open problem.

Open Challenges

- **Compositional Safety Verification.** Future defenses should operate beyond individual skills. e.g., through compatibility checks, information-flow analysis, or formal verification of skill compositions—to detect emergent risks that arise only at the composition level.
- **Lifecycle-Continuous Defense.** Rather than treating pre-deployment vetting and runtime monitoring as separate stages, defense architectures should propagate trust signals across the full skill lifecycle, closing the gap exploited by install-clean-but-runtime-malicious skills.

7 Skill Evaluation and Feedback

Evaluation closes the feedback loop of skill-based systems by determining whether skills are *well-formed*, *useful*, *safe*, and *effectively utilized*. We organize this space as a structured pipeline: from evaluating skill generation and intrinsic quality, to assessing their usage through retrieval and composition, and finally to measuring safety under deployment conditions.

7.1 Skill Generation Evaluation

The first stage evaluates whether generated skills generalize beyond their source tasks. Early studies (Wang et al., 2026a; Zhou et al., 2026b; Yu et al., 2025) test transfer by applying skills learned on one benchmark to related tasks, but are typically limited to narrow domains such as web navigation. To better assess

generalization, EvoSkills (Zhang et al., 2026c) introduces self-generated tasks, while SkillLearnBench (Zhong et al., 2026) provides the first dedicated benchmark for continual skill learning across real-world scenarios.

However, current evaluations remain limited. Skill transfer depends critically on the *distance* between tasks—whether a skill can be reused directly, requires adaptation, or must be regenerated—yet this relationship remains largely unmodeled. As a result, existing benchmarks provide only partial evidence of generalization.

7.2 Skill Quality Evaluation

Given generated skills, the next question is whether they are intrinsically useful. We distinguish *static quality* (correctness and completeness) from *dynamic quality* (whether agents benefit from the skill during execution). SkillsBench (Li et al., 2026c) shows that curated skills improve performance inconsistently, while self-generated skills often provide limited gains. SWE-Skills-Bench (Han et al., 2026) further reveals that many real-world software skills yield negligible improvements.

These results highlight a gap between skill design and actual utility. Existing benchmarks often evaluate whether a skill works in isolation, but not *when* and *why* it should be applied in practice.

7.3 Skill Retrieval and Composition Evaluation

Beyond intrinsic quality, skills must be correctly selected and composed at runtime. SkillCraft (Chen et al., 2026b) shows that reusable skill composition improves success rates while reducing token usage, demonstrating the importance of structured coordination. SRA-Bench (Su et al., 2026) further disentangles retrieval, incorporation, and downstream performance, revealing that current models are neither *relevance-aware* nor *need-aware*.

These findings suggest that effective skill usage depends not only on skill quality but also on accurate retrieval and coherent composition.

7.4 Skill Safety Evaluation

Finally, evaluation must account for safety under deployment conditions. HarmfulSkillBench (Jiang et al., 2026c) measures static harmfulness of skills, showing that unsafe skills can systematically degrade model alignment. STARS (Zhang et al., 2026b) introduces SIA-Bench to evaluate safety at invocation time, incorporating runtime context.

Despite these advances, reliable pre-deployment and lifecycle-aware safety evaluation remains largely unexplored, particularly under compositional and dynamic settings.

7.5 Summary

This section traced skill evaluation along four stages—**skill generation** (§7.1), **intrinsic quality** (§7.2), **retrieval and composition** (§7.3), and **safety** (§7.4). Across these stages, a consistent shift emerges: evaluation is moving from measuring skills as *isolated artifacts* toward measuring them as *context-dependent components*, where utility, applicability, and safety all depend on the agent, task, and runtime conditions in which a skill is used.

Despite this progress, three fundamental challenges remain:

- **Isolated metrics, contextual reality.** Generalization, correctness, and safety are typically measured on individual skills under static conditions, while real performance depends on task distance, retrieval accuracy, and composition coherence—dimensions that current benchmarks rarely capture jointly.
- **Missing models of skill applicability.** No principled framework characterizes *when* a skill should be applied, *how far* it can transfer, or *how much* it actually helps a given agent— leaving a persistent gap between skill design and observed utility.

- **Lifecycle-blind safety evaluation.** Safety benchmarks largely target static harmfulness or single-invocation contexts, with limited coverage of compositional risks and post-deployment behavior shifts.

Open Challenges

- **Dynamic, Context-Dependent Utility Evaluation.** Future benchmarks should measure skill utility as a function of agent, task, and context—reporting not only whether a skill is correct, but how much it improves end-to-end performance under realistic usage conditions.
- **End-to-End Lifecycle Evaluation.** Rather than evaluating generation, retrieval, composition, and safety in isolation, future frameworks should assess them jointly within a unified feedback loop, enabling principled diagnosis of where skill-based systems fail.

8 Skill-Native Systems and Ecosystems

As skills become a first-class abstraction, recent work increasingly treats them as the organizing principle of agent systems. We view this space as an evolving system stack: from **skill-centric platforms** that organize and manage skills, to **open ecosystems** that enable their sharing and evolution, and finally to **execution infrastructure** that supports portable and consistent skill deployment.

8.1 Skill-Centric Agent Platforms

The first stage focuses on organizing skills within individual systems. SkillNet (Liang et al., 2026b) introduces ontology-based structures to manage large-scale skill libraries across heterogeneous sources. GEMS (He et al., 2026) integrates skills into multi-agent loops, treating them as modular expert knowledge components.

These platforms establish skills as system primitives, highlighting the need for scalable organization, versioning, and interoperability. However, they primarily operate within isolated environments, with limited support for cross-agent sharing and evolution.

8.2 Open and Self-Evolving Agent Ecosystems

Building on platform-level organization, the next stage enables skills to be shared, reused, and evolved across agents. OpenClaw (OpenClaw Team, 2026) establishes a community-driven repository with standardized artifacts (e.g., `SKILL.md`), while OpenSpace (Huang & Team, 2026) introduces automated skill evolution and cross-agent transfer. EvoAgent (Zhang et al., 2026a) and RevFactory (revfactory, 2026) further integrate skill learning with multi-agent delegation and system-level allocation.

This shift transforms skills from static assets into dynamic, ecosystem-level resources. However, existing work primarily focuses on creation and sharing, leaving systematic library management underexplored: merging redundant skills, splitting overly broad ones, maintaining consistent versioning, and preserving global coherence remain open challenges.

8.3 Skill Execution Infrastructure

As skill ecosystems scale, an emerging challenge is ensuring that skills execute consistently across heterogeneous LLMs and agent frameworks. This line of work treats skills as portable executable units, introducing compilation and runtime abstractions for cross-platform execution.

SkVM (Chen et al., 2026a) pioneers this direction by modeling skills as executable programs and LLMs as heterogeneous execution backends. Inspired by language virtual machines, it formalizes skill execution through capability profiles and compilation models, enabling execution across different models and harnesses.

Building on this idea, SkCC (Ouyang et al., 2026b) extends skill portability into a full compiler pipeline with frontend parsing, an intermediate representation (SkIR), analysis, and backend emitters. A single `SKILL.md` artifact can thus be compiled into platform-specific formats optimized for different models. SkCC

further incorporates compile-time safety analysis through *Anti-Skill Injection*, automatically detecting risky execution patterns and injecting corresponding constraints into SkIR.

Overall, this direction elevates skills from reusable prompts to portable system-level components, focusing on compilation, adaptation, and secure execution across heterogeneous agent ecosystems.

8.4 Cross-Cutting Challenges and Summary

This section traced skill-native systems along an expanding stack: **skill-centric platforms** that organize skills within individual agents, **open ecosystems** that enable cross-agent sharing and evolution, and **execution infrastructure** that supports portable deployment across heterogeneous models and frameworks.

Across these layers, skills are evolving from agent-internal artifacts into shared system-level infrastructure. However, unlike models and datasets, skill ecosystems still lack mature standards, lifecycle management, and governance mechanisms. We identify three cross-cutting challenges that emerge across the entire stack.

Standardization vs. diversity. Portable execution infrastructures require stable interfaces and shared abstractions, while open ecosystems thrive on heterogeneous community contributions. Premature standardization risks limiting innovation, whereas insufficient standardization fragments interoperability across platforms and models.

Quality vs. openness. Open contribution naturally produces a long tail of low-quality or unsafe skills. Unlike traditional software repositories, poorly designed skills may not only fail individually but also introduce unpredictable behaviors when composed with other skills. This calls for ecosystem-level quality control mechanisms combining static analysis, usage-based evaluation, and compositional safety checks.

Lifecycle coherence across layers. Skills continuously evolve through refinement, versioning, forking, and deprecation, yet current systems provide weak support for propagating these changes across platforms and downstream agents. As a result, updated skills may silently break dependent workflows or remain inconsistent across ecosystems. Building lifecycle protocols for semantic skill evolution therefore remains an important open problem.

Overall, current skill ecosystems remain vertically fragmented: platforms, open repositories, and execution infrastructures are often developed in isolation, with limited support for cross-framework portability, global governance, and long-term maintenance.

Open Challenges

- **Ecosystem-Level Quality Curation.** Future registries should integrate usage-based scoring, compositional safety checks, and automated deduplication into a unified curation pipeline—moving beyond per-skill vetting toward library-level coherence.
- **Semantic Versioning for Skills.** Protocols for versioning, deprecation, and dependency tracking need to account for the semantic drift of natural-language descriptions, where a skill "update" may silently change behavior without altering its interface.
- **Cross-Layer Standardization.** Common abstractions and exchange protocols across platform, ecosystem, and execution layers are needed to prevent vertical fragmentation from becoming a permanent barrier to ecosystem maturation.

9 Conclusion

This survey reviewed the emergence of *skills* as a core abstraction for LLM agents across construction, access, training, safety, evaluation, and systems. Across these dimensions, skills are evolving from static prompts into dynamic, reusable, and system-level components. Three challenges repeatedly emerge: compositional interactions between skills, weak lifecycle management for evolving skill libraries, and the adaptability–efficiency trade-off between external skills and parameter internalization.

AI Usage Statement

We use AI-based language tools (e.g., ChatGPT) during manuscript preparation to assist with language polishing and expression refinement only. All research content, analyses, and conclusions were developed solely by the authors. The authors take full responsibility for the integrity and accuracy of the work presented.

References

Vivek Acharya. Skill poisoning: Attack taxonomies and defense architectures for composable agent skill ecosystems in ai-driven cyber-physical systems. *Available at SSRN 6408998*, 2026.

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.

John R Anderson. Acquisition of cognitive skill. *Psychological review*, 89(4):369, 1982.

Anthropic. Skills: Public Repository for Agent Skills. `https://github.com/anthropics/skills`, 2026. GitHub repository. Accessed: 2026-04-28.

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

Le Chen, Erhu Feng, Yubin Xia, and Haibo Chen. Skvm: Revisiting language vm for skills across heterogenous llms and harnesses, 2026a. URL `https://arxiv.org/abs/2604.03088`.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning. *Advances in Neural Information Processing Systems*, 37:589–631, 2024.

Shiqi Chen, Jingze Gai, Ruochen Zhou, Jinghan Zhang, Tongyao Zhu, Junlong Li, Kangrui Wang, Zihan Wang, Zhengyu Chen, Klara Kaleb, et al. Skillcraft: Can llm agents learn to use tools skillfully? *arXiv preprint arXiv:2603.00718*, 2026b.

Tianyi Chen, Yinheng Li, Michael Solodko, Sen Wang, Nan Jiang, Tingyuan Cui, Junheng Hao, Jongwoo Ko, Sara Abdali, Leon Xu, et al. Cua-skill: Develop skills for computer using agent. *arXiv preprint arXiv:2601.21123*, 2026c.

Cisco AI Defense. Skill scanner. GitHub, 2026. URL `https://github.com/cisco-ai-defense/skill-scanner`. Accessed: 2026.

Luis Miguel Vieira Da Silva, Aljosha Köche, Nicolas König, Felix Gehlhoff, and Alexander Fay. Capability-driven skill generation with llms: A rag-based approach for reusing existing libraries and interfaces. In *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8. IEEE, 2025.

Zenghao Duan, Yuxin Tian, Zhiyi Yin, Liang Pang, Jingcheng Deng, Zihao Wei, Shicheng Xu, Yuyao Ge, and Xueqi Cheng. Skillattack: Automated red teaming of agent skills through attack path refinement. *arXiv preprint arXiv:2604.04989*, 2026.

Zhiyuan Fan, Tinghao Yu, Yuanjun Cai, Jiangtao Guan, Yun Yang, Dingxin Hu, Jiang Zhou, Xing Wu, Zhuo Han, Feng Zhang, et al. Toward scalable terminal task synthesis via skill graphs. *arXiv preprint arXiv:2604.25727*, 2026.

Yunhao Feng, Yifan Ding, Yingshui Tan, Boren Zheng, Yanming Guo, Xiaolong Li, Kun Zhai, Yishan Li, and Wenke Huang. Skilltrojan: Backdoor attacks on skill-based agent systems. *arXiv preprint arXiv:2604.06811*, 2026.

Yudong Gao, Zongjie Li, Zimo Ji, Pingchuan Ma, Shuai Wang, et al. Skillreducer: Optimizing llm agent skills for token efficiency. *arXiv preprint arXiv:2603.29919*, 2026.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32, 2023.

Jingzhi Gong, Ruizhen Gu, Zhiwei Fei, Yazhuo Cao, Lukas Twist, Alina Geiger, Shuo Han, Dominik Sobania, Federica Sarro, and Jie M Zhang. Skillmoo: Multi-objective optimization of agent skills for software engineering. *arXiv preprint arXiv:2604.09297*, 2026.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models, 2024.

Zihan Guo, Zhiyu Chen, Xiaohang Nie, Jianghao Lin, Yuanjian Zhou, and Weinan Zhang. Skillprobe: Security auditing for emerging agent skill marketplaces via multi-agent collaboration. *arXiv preprint arXiv:2603.21019*, 2026.

Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. Swe-skills-bench: Do agent skills actually help in real-world software engineering? *arXiv preprint arXiv:2603.15401*, 2026.

Zefeng He, Siyuan Huang, Xiaoye Qu, Yafu Li, Tong Zhu, Yu Cheng, and Yang Yang. Gems: Agent-native multimodal generation with memory and skills. *arXiv preprint arXiv:2603.28088*, 2026.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Yinghan Hou and Zongyou Yang. Skillsieve: A hierarchical triage framework for detecting malicious ai agent skills. *arXiv preprint arXiv:2604.06550*, 2026.

Chao Huang and HKUDS Team. Openspace: A self-evolving skill engine for agent systems, 2026. URL `https://github.com/HKUDS/OpenSpace`. Accessed: 2026-04-27.

Chenyi Huang, Haoting Zhang, Jingxu Xu, Zeyu Zheng, and Yunduan Lin. Bilevel optimization of agent skills via monte carlo tree search. *arXiv preprint arXiv:2604.15709*, 2026.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 1051–1068, 2023.

Ken Huang and Jerry Huang. Audited skill-graph self-improvement for agentic llms via verifiable rewards, experience synthesis, and continual memory. *arXiv preprint arXiv:2512.23760*, 2025.

Xiaojun Jia, Jie Liao, Simeng Qin, Jindong Gu, Wenqi Ren, Xiaochun Cao, Yang Liu, and Philip Torr. Skillject: Automating stealthy skill-based prompt injection for coding agents with trace-driven closed-loop refinement. *arXiv preprint arXiv:2602.14211*, 2026.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, et al. Xskill: Continual learning from experience and skills in multimodal agents. *arXiv preprint arXiv:2603.12056*, 2026a.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. Sok: Agentic skills–beyond tool use in llm agents. *arXiv preprint arXiv:2602.20867*, 2026b.

Yukun Jiang, Yage Zhang, Michael Backes, Xinyue Shen, and Yang Zhang. Harmfulskillbench: How do harmful skills weaponize your agents? *arXiv preprint arXiv:2604.15415*, 2026c.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, Heather Miller, et al. Dspy: compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Dawei Li, Zongxia Li, Hongyang Du, Xiyang Wu, Shihang Gui, Yongbei Kuang, and Lichao Sun. Graph of skills: Dependency-aware structural retrieval for massive agent skills. *arXiv preprint arXiv:2604.05333*, 2026a.

Fangzhou Li, Pagkratios Tagkopoulos, and Ilias Tagkopoulos. Skillflow: Scalable and efficient agent skill retrieval system. *arXiv e-prints*, pp. arXiv–2504, 2025.

Hao Li, Chunjiang Mu, Jianhao Chen, Siyue Ren, Zhiyao Cui, Yiqun Zhang, Lei Bai, and Shuyue Hu. Organizing, orchestrating, and benchmarking agent skills at ecosystem scale. *arXiv preprint arXiv:2603.02176*, 2026b.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026c.

Yuyang Li, Yiran Dou, Jie-Jing Shao, Yueming Lyu, Ivor Tsang, and Haiyan Yin. Skilltracer: Structural failure attribution and refinement of agentic skills in long-horizon web tasks. In *Workshop on Multi-Agent Learning and Its Opportunities in the Era of Generative AI*, 2026d.

Qiliang Liang, Hansi Wang, Zhong Liang, and Yang Liu. From skill text to skill structure: The scheduling-structural-logical representation for agent skills. *arXiv preprint arXiv:2604.24026*, 2026a.

Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, et al. Skillnet: Create, evaluate, and connect ai skills. *arXiv preprint arXiv:2603.04448*, 2026b.

Xingyan Liu, Xiyue Luo, Linyu Li, Ganghong Huang, Jianfeng Liu, and Honglin Qiao. Skillforge: Forging domain-specific, self-evolving agent skills in cloud technical support. *arXiv preprint arXiv:2604.08618*, 2026a.

Yi Liu, Zhihao Chen, Yanjun Zhang, Gelei Deng, Yuekang Li, Jianting Ning, Ying Zhang, and Leo Yu Zhang. Malicious agent skills in the wild: A large-scale security empirical study. *arXiv preprint arXiv:2602.06547*, 2026b.

Yi Liu, Weizhe Wang, Ruitao Feng, Yao Zhang, Guangquan Xu, Gelei Deng, Yuekang Li, and Leo Zhang. Agent skills in the wild: An empirical study of security vulnerabilities at scale. *arXiv preprint arXiv:2601.10338*, 2026c.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*, 2026a.

Zijian Lu, Yiping Zuo, Yupeng Nie, Xin He, Weibei Fan, and Chen Dai. Contractskill: Repairable contract-based skills for multimodal web agents. *arXiv preprint arXiv:2603.20340*, 2026b.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, et al. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*, 2025.

Lijia Lv, Xuehai Tang, Jie Wen, Jizhong Han, and Songlin Hu. Structured security auditing and robustness enhancement for untrusted agent skills. *arXiv preprint arXiv:2604.25109*, 2026.

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.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022a.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022b.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

Zheng Nie, Ruolin Shen, Xinlei Yu, Bo Yin, Jiangning Zhang, and Xiaobin Hu. Skillgraph: Self-evolving multi-agent collaboration with multimodal graph topology. *arXiv preprint arXiv:2604.17503*, 2026.

OpenClaw Community. ClawHub: A public skill registry for agent skills. `https://github.com/openclaw/clawhub`, 2026. GitHub repository.

OpenClaw Team. Openclaw: Personal ai assistant. `https://github.com/openclaw/openclaw`, 2026. GitHub repository.

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. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, Maohao Shen, Vishy Tirumalashetty, George Lee, Jiawei Han, Tomas Pfister, and Chen-Yu Lee. Skillos: Learning skill curation for self-evolving agents, 2026a. URL `https://arxiv.org/abs/2605.06614`.

Yipeng Ouyang, Yi Xiao, Yuhao Gu, and Xianwei Zhang. Skcc: Portable and secure skill compilation for cross-framework llm agents. *arXiv preprint arXiv:2605.03353*, 2026b.

OWASP Foundation. OWASP agentic skills top 10. `https://owasp.org/www-project-agentic-skills-top-10/`, 2026. Version 1.0, Q2 2026.

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*, pp. 1–22, 2023.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

Cheng Qian, Chi Han, Yi Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 6922–6939, 2023.

Yubin Qu, Yi Liu, Tongcheng Geng, Gelei Deng, Yuekang Li, Leo Yu Zhang, Ying Zhang, and Lei Ma. Supply-chain poisoning attacks against llm coding agent skill ecosystems. *arXiv preprint arXiv:2604.03081*, 2026.

revfactory. Harness: A meta-skill for designing agent teams and skill architectures, 2026. URL `https://github.com/revfactory/harness`. Accessed: 2026-04-27.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

David Schmotz, Luca Beurer-Kellner, Sahar Abdelnabi, and Maksym Andriushchenko. Skill-inject: Measuring agent vulnerability to skill file attacks. *arXiv preprint arXiv:2602.20156*, 2026.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi GU, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning, 2026. URL `https://arxiv.org/abs/2605.06130`.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Snyk. ToxicSkills: Malicious AI agent skills in ClawHub. Snyk Blog, February 2026. URL `https://snyk.io/blog/toxicskills-malicious-ai-agent-skills-clawhub/`. Accessed: 2026.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, et al. Restgpt: Connecting large language models with real-world restful apis. *arXiv preprint arXiv:2306.06624*, 2023.

Weihang Su, Jianming Long, Qingyao Ai, Yichen Tang, Changyue Wang, Yiteng Tu, and Yiqun Liu. Skill retrieval augmentation for agentic ai, 2026. URL `https://arxiv.org/abs/2604.24594`.

Liran Tal. Why your "Skill Scanner" is just false security (and maybe malware). Snyk Blog, February 2026. URL `https://snyk.io/blog/skill-scanner-false-security/`. Accessed: 2026.

Yu Tian, Jiawei Chen, Lifan Zheng, Mingxiang Tao, Xinyi Zeng, Zhaoxia Yin, Hang Su, and Xian Sun. Skills-coach: A self-evolving skill optimizer via training-free grpo. *arXiv preprint arXiv:2604.27488*, 2026.

Guiyao Tie, Jiawen Shi, Pan Zhou, and Lichao Sun. Badskill: Backdoor attacks on agent skills via model-in-skill poisoning. *arXiv preprint arXiv:2604.09378*, 2026.

Songjun Tu, Chengdong Xu, Qichao Zhang, Yaocheng Zhang, Xiangyuan Lan, Linjing Li, and Dongbin Zhao. Dynamic dual-granularity skill bank for agentic rl. *arXiv preprint arXiv:2603.28716*, 2026.

Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, et al. Skillx: Automatically constructing skill knowledge bases for agents. *arXiv preprint arXiv:2604.04804*, 2026a.

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*, 2023a.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*, 2026b.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*, 2025.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 2609–2634, 2023b.

Zhaoyang Wang, Qianhui Wu, Xuchao Zhang, Chaoyun Zhang, Wenlin Yao, Fazle Elahi Faisal, Baolin Peng, Si Qin, Suman Nath, Qingwei Lin, et al. Webxskill: Skill learning for autonomous web agents. *arXiv preprint arXiv:2604.13318*, 2026c.

Zihan Wang, Rui Zhang, Yu Liu, Chi Liu, Qingchuan Zhao, Hongwei Li, and Guowen Xu. Black-box skill stealing attack from proprietary llm agents: An empirical study. *arXiv preprint arXiv:2604.21829*, 2026d.

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*, 2026e.

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

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.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026a.

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*, 2026b.

Wenjie Xiao, Xuehai Tang, Biyu Zhou, Songlin Hu, and Jizhong Han. Routeguard: Internal-signal detection of skill poisoning in llm agents. *arXiv preprint arXiv:2604.22888*, 2026.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, et al. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

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

Tik Yu Yim, Wenting Tan, Sum Yee Chan, Tak-Wah Lam, and Siu Ming Yiu. Asda: Automated skill distillation and adaptation for financial reasoning. *arXiv preprint arXiv:2603.16112*, 2026.

Simon Yu, Gang Li, Weiyan Shi, and Peng Qi. Polyskill: Learning generalizable skills through polymorphic abstraction. *arXiv preprint arXiv:2510.15863*, 2025.

Boqin Yuan, Renchu Song, Yue Su, Sen Yang, and Jing Qin. Clawtrace: Cost-aware tracing for llm agent skill distillation, 2026. URL `https://arxiv.org/abs/2604.23853`.

Aimin Zhang, Jiajing Guo, Fuwei Jia, Chen Lv, Boyu Wang, and Fangzheng Li. Evoagent: An evolvable agent framework with skill learning and multi-agent delegation. *arXiv preprint arXiv:2604.20133*, 2026a.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–20, 2025.

Guijia Zhang, Shu Yang, Xilin Gong, and Di Wang. Stars: Skill-triggered audit for request-conditioned invocation safety in agent systems. *arXiv preprint arXiv:2604.10286*, 2026b.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, et al. Evoskills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026c.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026d.

Guangxiang Zhao, Qilong Shi, Xusen Xiao, Xiangzheng Zhang, Tong Yang, and Lin Sun. Thinking with reasoning skills: Fewer tokens, more accuracy. *arXiv preprint arXiv:2604.21764*, 2026.

Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, et al. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, et al. Skillrouter: Skill routing for llm agents at scale. *arXiv e-prints*, pp. arXiv–2603, 2026.

Shanshan Zhong, Yi Lu, Jingjie Ning, Yibing Wan, Lihan Feng, Yuyi Ao, Leonardo FR Ribeiro, Markus Dreyer, Sean Ammirati, and Chenyan Xiong. Skillearnbench: Benchmarking continual learning methods for agent skill generation on real-world tasks. *arXiv preprint arXiv:2604.20087*, 2026.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, et al. Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering. *arXiv preprint arXiv:2604.08224*, 2026a.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, et al. Memento-skills: Let agents design agents. *arXiv preprint arXiv:2603.18743*, 2026b.

Jiaying Zhu, Lyuye Zhang, Wenbo Guo, and Yang Liu. Skillclone: Multi-modal clone detection and clone propagation analysis in the agent skill ecosystem. *arXiv preprint arXiv:2603.22447*, 2026.