What Should Agent Evaluation Measure When Agents Change Their Environments? A Survey of Agent-Environment Co-Evolution

Anonymous ACL submission

Abstract

When agents leave behind tools, memories, code, data, or policies, task success alone no longer characterizes evaluation. Reset protocols remain reproducible, but they discard inherited state and cannot measure whether an agent improves, degrades, or burdens the environment used by later agents. Using a mechanism-driven 133-reference corpus, we survey agent-environment co-evolution across benchmarks, environment design, tool and memory creation, software agents, synthetic-data feedback, multi-agent norms, self-evolving agents, and safety. We define ARTIFACT ECOLOGY as the inherited tools, memories, documents, tests, schemas, caches, policies, and governance records through which agent behavior persists, and organize the literature by feedback direction, artifact type, persistence mechanism, and co-evolutionary regime. This framing motivates reset, persistent, governed-persistent, and held-out inheritance conditions that preserve existing tasks and verifiers while measuring artifact visibility, reuse, governance, transfer, library growth, and cost. Lightweight coding checks and small benchmark probes instantiate these measurements, including a controlled ablation showing that more inherited state can increase context cost without improving task success. The analysis motivates agent-evaluation protocols that report both task outcomes and persistent environmental effects.

1 Introduction

Modern AI evaluation often fixes the input, resets context, and compares outputs with a fixed target. This template enabled broad capability suites and holistic scorecards (Hendrycks et al., 2021; Srivastava et al., 2023; Liang et al., 2023). Agentic frameworks strain the assumption: they run workflows through tools, memory, code repositories, documents, databases, policies, and user-facing systems (Yao et al., 2023; Shinn et al., 2023; Wu et al., 2024; Hong et al., 2024; Chen et al., 2025b). As agents move into interactive settings, the evaluated unit is increasingly a *system*, not only a model call (Wang et al., 2024b; Xi et al., 2023; Guo et al., 2024).

Agent benchmarks have expanded from text examples to multi-turn tasks grounded in websites, desktops, tools, and repositories (Liu et al., 2024; Ma et al., 2024; Zhou et al., 2024; Koh et al., 2024; Xie et al., 2024; Drouin et al., 2024; Jimenez et al., 2024). Many still report per-task success after each environment reset. This supports reproducibility, but becomes incomplete when coding agents edit repositories (Jimenez et al., 2024; Yang et al., 2024), tool agents create reusable skills (Qian et al., 2023; Wang et al., 2023), multi-agent systems form norms (Park et al., 2023; Gao et al., 2025), and model outputs enter future corpora (Shumailov et al., 2024; Chen et al., 2025c).

Environments are also sources of capability: world models, curricula, generated worlds, and environment distributions shape what agents learn (Dennis et al., 2020; Parker-Holder et al., 2022; Huang et al., 2025; Chen et al., 2025a; Wang et al., 2026). Recent work goes further: agents can write tools, memories, code, data, and rules that become the environment for later agents (Luo et al., 2025; Zhang et al., 2026a; Yang et al., 2026; Ji et al., 2026; Jiang et al., 2026; Fang et al., 2026).

We study this feedback as **agent-environment co-evolution**: agents adapt to environments while their actions modify the harness later agents face. Existing work traces related patterns in game agents, synthetic-data feedback, agent societies, tool and skill libraries, self-evolving agents, and environment scaling (Baker et al., 2020; Czarnecki et al., 2020; Shumailov et al., 2024; Alemohammad et al., 2024; Park et al., 2023; Gao et al., 2025; Bracale Syrnikov et al., 2026; Wang et al., 2023; Qian et al., 2023; Huang et al., 2026b, 2025; He et al., 2026; Yang et al., 2026; Zhang et al., 2026b). What remains underdeveloped is how to measure

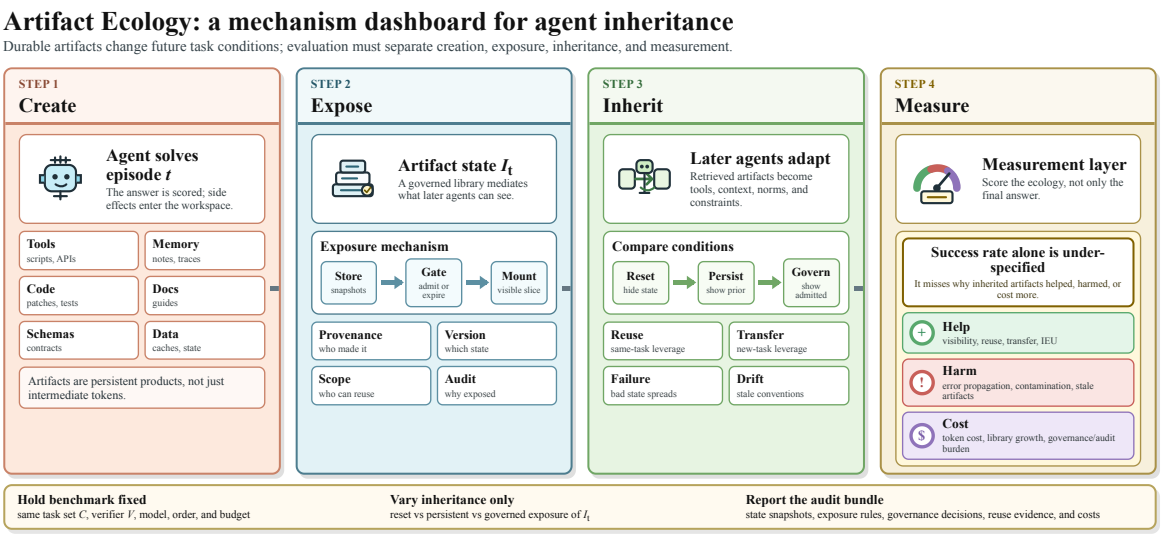

Figure 1: ARTIFACT ECOLOGY as an inheritance-aware measurement layer. Agents create durable artifacts; exposure and governance determine what later agents inherit; later agents adapt to that inherited state; evaluation must report not only success rate, but also help, harm, and cost through visibility, reuse, transfer, error propagation, library growth, governance, and audit burden.

the environment agents leave for later agents.

This creates an evaluation mismatch. Static benchmarks measure isolated task success, and agent benchmarks extend task success to interactive settings (Hendrycks et al., 2021; Srivastava et al., 2023; Liang et al., 2023; Liu et al., 2024; Ma et al., 2024; Zhou et al., 2024; Koh et al., 2024; Xie et al., 2024; Jimenez et al., 2024). They do not measure whether a system leaves useful tools, misleading memories, brittle wrappers, contaminated data, or governance burden. The missing object is the *artifact ecology* created by agent behavior.

We call this ARTIFACT ECOLOGY: the tools, memories, documents, tests, schemas, caches, policies, and governance records that mediate inheritance. Static benchmarks need not be discarded; they can instead be reorganized into reset, persistent, governed-persistent, and held-out inheritance conditions that expose downstream benefit or harm, reuse, governance, error propagation, and cost. Executable probes illustrate these fields by making score shifts, admission and rejection traces, and transfer failures reportable within one protocol.

For deployed agents, the artifact left after a task is part of the outcome. Evaluation therefore has to measure not only answers, but also the environments agents leave behind. Figure 1 summarizes this measurement layer.

Scope of claims. This paper makes a survey-plus-protocol claim: it identifies inherited state across agent literatures, specifies what a benchmark wrapper should preserve and report, and uses executable probes only as auditability checks. The survey follows four prose questions: what agents inherit, how they change it, which regimes emerge, and what evaluation should preserve.

Relation to adjacent surveys. Adjacent surveys explain how agents are built and assessed: architectures, planning, memory, tool use, coordination, environment scaling, self-evolving loops, safety, and trust (Wang et al., 2024b; Xi et al., 2023; Guo et al., 2024; Huang et al., 2025; Luo et al., 2025; Zhang et al., 2026a; Yuan et al., 2024; Gu et al., 2025). They are complementary, but most treat memory, tools, code, data, norms, and policies as separate design or risk categories. Our organizing variable is whether state can persist, be inherited, and change later behavior. On that axis, a tool library, a stale memory, a repository patch, synthetic-data contamination, an emergent norm, and an admission policy become instances of the same evaluation problem. Appendix Table 3 gives the survey-by-survey distinction.

Contributions. We make four contributions.

1. **Inherited environment as evaluation target.** We synthesize a 133-reference corpus showing

that persistent inherited state recurs across agent systems but is removed by reset evaluation (§2–§3).

2. **Artifact Ecology.** We define the inherited tools, memory, documents, tests, schemas, caches, policies, and governance metadata through which agent actions persist, and we state boundary conditions for when such state should count as an evaluation object.

3. **Inheritance-aware protocols and metrics.** We derive reset, persistent, governed-persistent, and held-out inheritance conditions, with metrics for visibility, reuse, governance, transfer, error propagation, library growth, and cost (§4).

4. **Diagnostic and controlled instantiation.** We provide a task-pool construction procedure, diagnostic studies, and a fixed-slice artifact-library ablation showing that inherited environments expose variables reset evaluation hides. The probes support measurement and stress-test claims, not a general causal claim that persistence improves agent competence (§5).

2 Survey Evidence: Agents Shape and Inherit Environments

The survey evidence is organized around four questions an inheritance-aware evaluator must answer. The corpus covers work in which environment state persists across episodes or generations and can shape later behavior. We include environment→agent, agent→environment, and coupled feedback paths, while excluding one-step inputs, purely episodic tool or memory use, and architecture surveys without an inheritable environment variable. Starting from agent benchmarks, self-play, environment design, model collapse, open-ended learning, safety, software agents, and evaluation methods, we expand through citation following, adjacent-survey comparison, and targeted keyword search; the snapshot was frozen on May 25, 2026.

Each work is coded by feedback direction, affected variable, persistence mechanism, and dominant regime. The corpus is mechanism-driven: it aims to cover distinct forms of inheritance rather than estimate prevalence. Appendix Figure 2 gives the full corpus mechanism map, and Appendix Table 5 states the construction, coding schema, and consistency checks.

2.1 What do agents inherit?

Agents inherit more than prompts. Foundation-model evaluation made progress measurable by fixing inputs, contexts, and targets (Hendrycks et al., 2021; Srivastava et al., 2023; Liang et al., 2023); behavioral and dynamic evaluation showed that static held-out accuracy can miss targeted failures and benchmark artifacts (Ribeiro et al., 2020; Kiela et al., 2021). Agent benchmarks extend the substrate to websites, desktops, tools, customer-service workflows, function calls, and software repositories (Liu et al., 2024; Ma et al., 2024; Zhou et al., 2024; Koh et al., 2024; Xie et al., 2024; Drouin et al., 2024; Jimenez et al., 2024; Yan et al., 2024). In these settings the benchmark is already an environment: it supplies affordances, constraints, verifiers, and sometimes mutable state.

Agents also inherit capability sources. Niche construction, evolutionary games, and open-ended evolution provide vocabulary for inheritance, cycling, arms races, and novelty (Odling-Smee et al., 2003; Maynard Smith, 1982; Stanley et al., 2019). Recent AI work operationalizes these ideas through self-play, unsupervised environment design, autocurricula, environment scaling, self-evolving loops, and world models (Baker et al., 2020; Dennis et al., 2020; Parker-Holder et al., 2022; Huang et al., 2025; Chen et al., 2025a; Wang et al., 2026; Luo et al., 2025; Zhang et al., 2026a). For this survey, inherited state becomes evaluation-relevant only when it can persist beyond the episode, be visible or retrievable later, plausibly change behavior, cost, safety, or governance burden, and be audited. That boundary excludes ordinary scratchpads and discarded one-off tool calls.

Implication.

Reset benchmarks are not invalid; they answer a narrower question. The first inheritance-aware measurement question is what the reset removed: which tools, memories, code, data, norms, policies, or governance records could later agents have inherited?

2.2 How do agents reshape future environments?

Agents turn task actions into future conditions. Embodied and virtual agents move objects, mine resources, and construct worlds (Wang et al., 2023; Fan et al., 2022; Cai et al., 2024a); tool and skill agents create reusable APIs, code, skill libraries, and tool memories (Qian et al., 2023; Cai et al., 2024b; Huang et al., 2026b); software agents mod-

ify repositories, tests, conventions, and workflows (Yang et al., 2024; Li et al., 2026; Khan et al., 2026). Other inherited environments are social or informational: routines, norms, markets, communication protocols, institutions, generated text or code, search traces, benchmark refreshes, and mutable information arenas (Park et al., 2023; Gao et al., 2025; Bracale Syrnikov et al., 2026; Shumailov et al., 2024; Chen et al., 2025c; Han et al., 2025; Ji et al., 2026). Across domains, outputs become conditions for later behavior.

The main structural distinction is not physical versus digital, but reversible versus hard to reverse. Reversible changes, such as moved objects or temporary environment configurations, permit correction and exploration. Hard-to-reverse changes, such as published data, repository dependencies, contaminated corpora, or deployed rules, create path dependence. Governance state is part of that path: admission rules, rejection logs, deprecations, permissions, and audit traces can protect later agents, but they also add review cost and institutional burden. Thus a useful artifact and a costly artifact may look identical to a reset score.

Implication.

Agent-written state should be evaluated as environmental inheritance, not as a task-solving byproduct. Reports should distinguish reversible from hard-to-reverse artifacts and should count governance burden alongside downstream usefulness.

2.3 When does feedback accumulate, race, transition, or collapse?

Once agents inherit environments and modify what will be inherited next, the object is the coupled trajectory. Positive feedback can accumulate tools, skills, and cooperation (Wang et al., 2023; Huang et al., 2026b; Bornemann et al., 2023; Wu et al., 2026); when bounded by tests, diversity, institutions, or resource limits, such feedback can support useful growth. Competitive coupling can instead produce Red Queen dynamics and adversarial adaptation, where each improvement is countered by a later response (Baker et al., 2020; Bansal et al., 2018; Czarnecki et al., 2020; Kumar et al., 2026). Threshold effects add another risk: phase transitions can turn gradual changes in context quality, communication, or task correlation into abrupt amplification or collapse (Liu et al., 2026).

Collapse findings are diagnostic because they recur across otherwise different systems. Model collapse and self-consuming data loops show distributional narrowing when generated data feeds future training data (Shumailov et al., 2024; Alemohammad et al., 2024; Dohmatob et al., 2024). Entropy collapse shows how training feedback can narrow policy exploration (Cui et al., 2025). Commons failures and degenerating information environments show how individually useful actions can degrade shared state (Bracale Syrnikov et al., 2026; Huang et al., 2026a; Mead et al., 2026; Mieczkowski et al., 2025). The common structural pattern is dominant positive feedback, hard-to-reverse modification, and insufficient diversity; the survey does not claim these are the only causes, but they are recurring variables evaluation should preserve.

Implication.

The same inheritance channel can produce accumulation, arms races, phase transitions, or collapse. Bounded positive feedback is productive; unbounded, hard-to-reverse feedback with low diversity is a warning sign that endpoint scores alone will miss.

2.4 What must evaluation preserve?

The evaluation requirement is counterfactual: with the same task source and native verifier, what changes when later agents inherit earlier artifacts? Static reset remains the isolated-competence anchor, but it should be paired with persistent streams, governed persistence, and held-out inheritance tests. A report should preserve the artifacts themselves, the exposure rule, the update rule, the governance rule, task order, verifier outputs, reuse evidence, costs, and enough transcript or trace material to audit whether visible artifacts were used.

Safety makes this preservation requirement concrete. Agentic workflows can become unsafe even when base models are aligned, because tools, memory, permissions, and policies introduce new states and attack surfaces (ClawSafety Team, 2026; Yuan et al., 2024; Gu et al., 2025; Lin et al., 2023; Pan et al., 2025; Xiang et al., 2025). Attacks and defenses co-adapt through prompt manipulation, toolchain attacks, sponge attacks, memory poisoning, and adaptive risk learning (Xu et al., 2024; Li et al., 2025; Li and Wang, 2026; Sharma et al., 2025; Wang et al., 2025; Xu et al., 2025; Pan et al., 2026). Governance decisions are therefore part of the evaluated environment rather than implementation detail. The diagnostic probes later in the paper instantiate these fields conservatively, as measurement and auditability checks rather than causal estimates of persistence improving competence.

Implication.

For persistent agents, the evaluation target expands from *Can the agent solve the task?* to *What environment does solving create, who inherits it, what is admitted or rejected, and what does that inheritance do to later agents?*

3 Taxonomy of Inherited Environment State

The survey evidence points to a concrete yet underexplored evaluation topic: inherited environment state. This broader object includes any persistent condition that later agents may face. We use ARTIFACT ECOLOGY for the narrower auditable protocol layer: artifacts through which earlier behavior changes later conditions. The category is broad but bounded: an artifact must persist beyond its episode, be visible or retrievable, be capable of changing later behavior, cost, safety, or governance burden, and be auditable. Table 1 organizes the main types. Their mechanisms differ, but reset evaluation misses the same effect: what the solution leaves for future work.

This boundary excludes within-episode scratchpads, transient planner states, and one-off tool calls or retrieval results discarded before later inheritance. Across inherited-state types, evaluation must record what persists, whether later agents see or use it, whether it transfers, propagates errors, and governance admits, rejects, or deprecates it.

Feedback regimes. The taxonomy describes what persists; the regime map describes what persistence does over time. The same inherited state can support accumulation, trigger arms races, cross thresholds, or collapse: accumulation adds reusable state, arms races couple competing adaptations, threshold shifts mark nonlinear changes after enough state or agents accumulate, and collapse degrades the inherited environment. Evaluation must measure artifacts, reuse, transfer, errors, governance, and cost, not only task scores.

Bridge to protocol. Appendix Table 7 expands the regime map. The rest of the paper turns these variables into a protocol: what to preserve, expose, govern, and score.

4 Artifact Ecology: Protocols and Metrics for Inherited Environments

The previous sections identify the state reset evaluation removes. This section derives ARTIFACT ECOLOGY as a protocol layer for evaluating agents together with the artifacts they leave behind, not as a standalone benchmark.

Evaluation lineage. Existing evaluation has moved beyond narrow static scores. Foundation-model, behavioral, dynamic, and interactive agent benchmarks broaden fixed-input testing into targeted tests, changing data, websites, desktops, tools, and repositories (Hendrycks et al., 2021; Srivastava et al., 2023; Liang et al., 2023; Ribeiro et al., 2020; Kiela et al., 2021; Liu et al., 2024; Ma et al., 2024; Zhou et al., 2024; Koh et al., 2024; Xie et al., 2024; Drouin et al., 2024; Jimenez et al., 2024). They expand the setting, but still leave one variable mostly unmeasured: state produced by earlier agents and inherited by later agents.

Artifact Ecology. Existing benchmarks already measure important fragments of this space: tool and function-calling benchmarks test use of available APIs (Schick et al., 2023; Qin et al., 2024; Patil et al., 2023; Yan et al., 2024); memory and reflection work studies state that can improve later reasoning (Shinn et al., 2023); long-horizon and dynamic benchmarks expose trajectories rather than isolated examples (Kiela et al., 2021; Fang et al., 2026; Jiang et al., 2026); and contamination studies track how generated content changes future evaluation material (Chen et al., 2025c; Han et al., 2025). ARTIFACT ECOLOGY puts artifact creation, exposure, reuse, governance, transfer, and cost under one reporting protocol.

It is not a replacement benchmark or leaderboard. It keeps existing tasks and verifiers, but changes the protocol by preserving selected artifacts, recording later usage, and comparing reset, persistent, governed, and held-out inheritance conditions. The category remains bounded by persistence, accessibility, consequence, and auditability.

4.1 Formalizing Agent-Environment Co-Evolution

In a reset benchmark, artifacts created in episode i cannot affect episode $i + 1$, matching the usual assumption that the environment is fixed during evaluation or learning (Sutton and Barto, 2018). Once outputs persist, reset evaluation removes the state through which downstream effects occur.

We use agent-environment co-evolution for the endogenous setting in which agent behavior changes an inheritable part of the environment and later agents adapt to that change. A state change is evaluation-relevant only if it persists beyond the

Inherited state type	What persists	How it is inherited	What reset evaluation misses
Tools and skills	APIs, scripts, wrappers, skill libraries, tool memories.	Later agents call, import, compose, or imitate prior tools.	Reuse, dependency debt, brittleness, and tool-library growth.
Memory and documentation	Notes, summaries, plans, reports, task cards, repository instructions.	Later agents retrieve or read prior written state.	Stale facts, misleading summaries, useful conventions, and context cost.
Code and workspace state	Repositories, tests, schemas, configuration, files, database migrations.	Later agents inherit modified workspaces or codebase conventions.	Technical debt, regression risk, schema drift, and maintainability.
Data and benchmark traces	Generated text, search traces, refreshed tasks, synthetic corpora, contamination records.	Future models, agents, or evaluators consume modified information environments.	Contamination, model collapse, distribution drift, and benchmark mutability.
Social and normative state	Roles, routines, communication protocols, market conventions, institutions, when represented in auditable artifacts.	Later agents interact under documented roles, routines, protocols, logs, or policies.	Coordination gains, collusion, norm drift, and institutional dependence.
Policy and governance state	Admission rules, rejected artifacts, permissions, safety policies, audit logs.	Later agents operate under accepted, blocked, or deprecated artifacts and rules.	Governance burden, reversibility, safety drift, and accountability.

Table 1: Taxonomy of inherited environment state. Artifact Ecology is not any state change; it is the persistent, accessible, consequential, and auditable subset that later agents can inherit. Latent social expectations count only when represented through inspectable roles, routines, protocols, logs, or policies.

creating episode, is accessible later, can plausibly change behavior, cost, safety, or governance burden, and can be audited. This narrower use excludes ordinary within-episode RL transitions and discarded scratch memory.

Definition 1 (Agent-Environment Co-Evolution)**.** Let t index an evaluation episode or stream step, not necessarily a model-training step. Let π_t denote the agent policy used at step t, τ_t the trajectory generated by that policy in the environment, and $\mathcal{M}_t = \langle \mathcal{S}_t, \mathcal{A}_t, T_t, R_t, \mathcal{I}_t \rangle$ denote the environment at step t, where $\mathcal{I}_t$ is an inheritable artifact state such as tools, memory, code, documents, data traces, schemas, policies, or governance records. An **inheritance-relevant co-evolutionary system** is a coupled update rule:

$$\pi_{t+1} = f(\pi_t, \mathcal{M}_t) \tag{1}$$

$$\mathcal{M}_{t+1} = g(\mathcal{M}_t, \tau_t), \quad \mathcal{I}_{t+1} \neq \mathcal{I}_t \tag{2}$$

where f is the agent adaptation operator (e.g., policy gradient, in-context learning, memory retrieval, or policy selection conditioned on inherited state) and g is the environment modification operator induced by the episode trajectory (e.g., tool creation, code or document changes, persistent data generation, norm emergence, or governance updates). A later agent may be a later run of the same frozen model with different inherited context, or a different policy or model. For fixed-weight LLM agents, f can be non-trivial when behavior changes through retrieval, memory, prompt context, or policy selection conditioned on inherited artifacts. The system is co-evolutionary for the purposes of this paper when **both** f and g are non-trivial, and when the non-trivial part of g modifies an inherited state $\mathcal{I}$ that satisfies the persistence, accessibility, consequence, and auditability conditions above.

If g is identity on inherited state, evaluation is static with respect to Artifact Ecology; if f is trivial, the environment drifts without agent adaptation. The setting of interest has both: agents write artifacts, later agents adapt to them, and the loop may accumulate capability, debt, arms races, or collapse. Three properties determine what evaluation must record: feedback polarity, reversibility, and timescale separation (Holland, 1992; Mitchell, 2009; Odling-Smee et al., 2003; Hofbauer and Sigmund, 1998).

4.2 Why Reset Scores Are Incomplete for Persistent Agents

Reset scores compare isolated competence, but cannot directly score frameworks whose effects include tools, memory, caches, generated code, or governance. Such systems may leave reusable tools, stale memory, brittle schemas, contaminated traces, or governance burden. If the benchmark resets after each task, those effects are discarded.

The mismatch has three forms. **Reset erases inheritance**: artifacts that help or harm later agents are removed. **Endpoint scores hide externalities**: equal pass rates can leave unequal debt, stale memory, brittle schemas, or governance burden. **Benchmarks become mutable**: generated code and search-time answer discovery can contaminate later evaluation environments (Chen et al., 2025c; Han et al., 2025; Ji et al., 2026).

Implication.

When artifacts persist, task success is not a complete unit of evaluation. The evaluated object is the coupled pair: agent policy plus inherited artifact ecology.

Mutable benchmark settings. Recent work gets closest when the benchmark itself changes: dynamic code evaluation and search-time contamination refresh or contaminate test material (Chen et al., 2025c; Han et al., 2025); ClawArena and

AgentLongBench introduce changing information environments and long rollouts (Ji et al., 2026; Fang et al., 2026); SEA-Eval targets self-evolving agents beyond episodic assessment (Jiang et al., 2026). These efforts move from fixed items to trajectories. Artifact Ecology adds artifact accounting: what persisted, who inherited it, what was admitted or rejected, and whether inheritance helped or harmed later agents.

Relation to closest protocols. Prior evaluation families often preserve one trajectory channel: changing data, long-horizon interaction, available tools, memory, or repository patches. The contribution is to require these fields jointly under fixed task and verifier anchors: what was preserved, exposed, used, admitted or rejected, whether it transferred, and what it cost. Appendix Table 4 gives the detailed comparison.

4.3 Design Requirements for Inheritance-Aware Agent Benchmarks

The synthesis motivates benchmark design requirements. Agentic benchmarks should still report task success, but also preserve and measure the environment that task solving creates, extending behavioral tests, dynamic benchmarks, agent benchmarks, contamination studies, and long-horizon rollouts (Ribeiro et al., 2020; Kiela et al., 2021; Liu et al., 2024; Ma et al., 2024; Chen et al., 2025c; Han et al., 2025; Fang et al., 2026). If artifacts can help, harm, or burden later agents, a benchmark should expose five quantities.

1. **Inheritance.** At least one condition should preserve artifacts produced by earlier episodes.
2. **Downstream usefulness.** Benchmarks should separate current-task success from the usefulness of the environment left for later agents.
3. **Transfer.** Inherited artifacts should be tested on held-out tasks, not only on the stream that produced them.
4. **Governance.** Admission, rejection, deprecation, tests, and policies should be measured as environment variables.
5. **Cost and debt.** Artifact libraries can save effort, consume context, grow complexity, or propagate errors; benchmark reports should make these effects visible.

Artifact Ecology operationalizes these requirements while keeping existing benchmark tasks and verifiers intact.

4.4 Artifact Ecology Conditions and Metrics

ARTIFACT ECOLOGY specifies task streams in which agents may answer, reuse, create, modify, or deprecate artifacts, and write tests or documentation. The environment is the artifact library plus exposure, admission, and deletion rules.

Operationally, the framework wraps a base benchmark. Let $\mathcal{B} = (C, V)$, where C is the task collection and V the native verifier. The wrapper adds an artifact space $\mathcal{A}$, an exposure rule h_t, an update rule u_t, and a governance rule q_t that admits, rejects, or deprecates candidates. A protocol condition fixes (h_t, u_t, q_t) while keeping C, V, model, task order, and budgets fixed as far as possible. Inherited state becomes measurable without changing the benchmark's original success criterion; reset remains the anchor.

We use four conditions: `static_reset` discards artifacts after each task; `persistent` exposes prior artifacts; `persistent_governed` exposes only admitted artifacts; and `inheritance_eval` tests held-out tasks in clean versus inherited environments. Appendix Table 9 summarizes the protocol conditions.

Let $S(C, E)$ denote benchmark score on collection C in environment E. The **static-to-persistent gap** is

$$\Delta_{\text{persist}} = S(C, E_{\text{persistent}}) - S(C, E_{\text{static}}),$$

and the **held-out inheritance contrast** is

$$\Delta_{\text{heldout}} = S(C_{\text{heldout}}, E_{\text{inherited}}) - S(C_{\text{heldout}}, E_{\text{clean}}),$$

where $E_{\text{inherited}}$ is produced by previous agents. This diagnostic contrast is not an estimate of artifact utility without stronger causal controls such as randomized exposure or matched-context ablations. We also report reuse, created/admitted/rejected/deprecated artifacts, error propagation, library size, and cost delta. Tools emphasize reuse and dependency debt; memory emphasizes staleness and context cost; code emphasizes regression risk; data traces emphasize contamination; governance emphasizes admission, rejection, deprecation, and audit cost. These metrics separate *solving* from *shaping*.

For executable studies, **artifact visibility** is the fraction of episodes in which inherited artifacts are available. **Artifact reuse** is the fraction in which

Probe	Diagnostic result
Claw-Bench	**Score:** reset 9/95 (9.5%), persistent 16/95 (16.8%), $\Delta = +7.4$ pp. **Artifact:** 100.0% visibility, 2.1% explicit reuse. **Read:** low reuse prevents treating the score gap as artifact utility.
BFCL notes	**Score:** reset 0.0%, persistent 5.0%, path-note score $\Delta = +0.018$. **Artifact:** 95.0% visibility, 45.0% explicit reuse. **Read:** inherited-note exposure is measurable even in a stateless function-call slice.
Library size	**Score:** fixed 24-task slice gives 9/6/8/9/8/9 passes for 0/5/10/25/50/100 target artifacts. **Artifact:** reuse 0.0–8.3%; largest setting raises mean input from 19.6k to 26.7k tokens. **Read:** more inherited state adds context cost without monotonic gains.

Table 2: Core diagnostic measurements under inherited-state wrappers. The library-size ablation fixes task order, model, verifier, and no-writer policy; the 100-target condition materializes 66 available candidates. Protocol details are in Appendix E.

the trajectory explicitly cites an inherited note, calls an inherited tool, or bases a tool/path choice on inherited content. **Admission** and **rejection** count governance decisions; rejected artifacts are logged but not exposed. These are artifact-level definitions: a task can be solved without reuse, and an artifact can be visible without being useful.

An Artifact Ecology should include task order, exposure and update rules, governance rule, artifact snapshots, admission/rejection logs, reuse labels, transcripts, verifier outputs, and cost records. The minimal schema covers the base task collection and verifier, artifact classes, exposure rule, update rule, governance rule, reuse evidence, held-out inheritance construction, and cost/library-growth accounting. Appendix Table 8 gives the checklist. Without it, inherited-environment claims are hard to distinguish from context or order effects.

5 From Survey Claims to Diagnostic Probe Results

The probes are diagnostic only: they test whether inherited-state variables can be observed on public benchmark material, not whether persistence improves competence. They use three concrete artifact forms: BFCL path notes that record API or argument patterns, Claw-Bench workspace artifacts such as helpers, tests, notes, or edited files, and preseeded libraries frozen under a no-writer policy. Artifacts are visible when exposed through the prompt or workspace; they count as reused only when the transcript explicitly cites, imports, calls, or follows them. Appendix E gives the protocol and audit details.

Main takeaways. Table 2 separates score movement from artifact evidence. Claw-Bench shows why this matters: the persistent score is higher, but reuse is only 2.1%, so the gap is a contrast rather than evidence of artifact utility. BFCL shows that inherited-note visibility and reuse can be measured even in a stateless function-call benchmark, while the library-size ablation shows that more inherited state can add context cost without monotonic pass-rate gains. Stronger empirical claims would require repeated runs, order perturbations, matched-context controls, confidence intervals, and independent validation of reuse labels.

Implication.

Executable Artifact Ecology probes make variables observable that reset benchmarks discard: static-to-persistent gap, artifact growth, governance admission/rejection, a limited held-out transfer sanity check, library-size effects, and memory visibility without task gains. The current numbers are diagnostic measurements; their role is to show what a stronger benchmark should control and report.

6 Conclusion

ARTIFACT ECOLOGY addresses a blind spot in agent evaluation: when agents leave behind tools, memories, code, documents, data, norms, or policies, the environment becomes part of the outcome future agents inherit. Reset benchmarks remain useful competence anchors, but they cannot distinguish capability from downstream usefulness, debt, contamination, governance burden, or cost. The survey and diagnostic probes therefore support a measurement claim rather than a leaderboard claim: agent-system evaluation should report both per-task performance and the quality of the inherited environment that performance creates.

Limitations & Future Work Directions

This survey is subject to several limitations. First, the literature on agents that shape and inherit their environments is expanding quickly, and our mechanism-driven corpus may miss recent or domain-specific work, especially in robotics, artificial life, HCI, and organizational studies. Second, the survey prioritizes recurring inheritance mechanisms over prevalence estimates; it should therefore be read as a synthesis of evaluation-relevant patterns, not as a systematic review of every paper in each area. Third, the cross-disciplinary concepts we use, including niche construction, complex adaptive systems, and governance, are orga-

nizing lenses rather than evidence that all agent systems follow the same dynamics. Finally, our protocol discussion is most concrete for tool-use, memory, software, and benchmark settings; embodied and low-level-control agents may require different artifact types, persistence mechanisms, and verifiers.

These limitations are also design constraints for the next generation of inherited-state evaluation. They point to a research agenda that keeps current task and verifier anchors, while making inheritance causal, governed, trajectory-aware, and comparable across domains.

Task streams as the evaluation unit. Deployed agents solve task streams while changing the state later tasks inherit, yet most benchmarks still reset each episode. The missing object is the artifact lineage: which tools, memories, code changes, tests, schemas, data traces, policies, and permissions were created, exposed, reused, hidden, or removed. Benchmarks should retain reset anchors while adding persistent, governed-persistent, and held-out branches with versioned snapshots and replayable task orders. This separates task solving from environment shaping: equal pass rates can leave environments with different downstream usefulness, cost, and fragility.

Causal estimands for inherited state. Once artifacts are visible, score changes conflate availability, use, marginal utility, order effects, and context cost. The missing object is the counterfactual effect of inherited state on later decisions or verifier outcomes. Protocols should include matched-context resets, randomized exposure, artifact ablations, order perturbations, seeded-library curves, and negative-transfer tasks. This distinguishes capability from correlation: a library matters only when it changes behavior beneficially, transferably, and with justified cost.

Governance and reversibility as benchmark variables. As artifacts become infrastructure, safety shifts from output filtering to environmental admission control. The missing object is the governance trajectory: provenance, validation, permission scope, admission or rejection, deprecation, rollback, and downstream behavior. Evaluations should score these as state transitions, including cases where harmful memories, brittle helpers, contaminated traces, or overbroad policies are accepted and reused. Safety is then measured where agent risk often materializes: in the persistence channel that determines what later agents may trust.

Trajectory-level regimes, not endpoint scores. At longer horizons, inherited state can accumulate, race, shift phase, or collapse while sharing the same short-run score. The missing object is the trajectory: artifact diversity, dependency depth, reversibility, error propagation, context-cost curves, adversarial histories, and thresholds where additional agents or artifacts change downstream value. Stress tests should vary horizon length, library growth, environment diversity, and governance strength while holding task sources and verifiers fixed. This lets benchmarks estimate when feedback remains productive, saturates, or becomes unstable.

Cross-domain inheritance with domain-specific verifiers. The same inheritance problem appears in tool use, software, embodied environments, multi-agent societies, synthetic-data feedback, and benchmark maintenance, but artifact types and failure modes differ. The missing object is a common reporting layer for physical layout state, repository state, data provenance, social norms, world-model divergence, and policy state, without assuming a single verifier fits all domains. Progress requires domain-local schemas mapped onto shared axes: persistence, visibility, reuse, transfer, reversibility, governance, and cost. This supports comparison between general co-evolutionary mechanisms and environment-specific artifacts.

References

Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, and 1 others. 2024. Self-consuming generative models go MAD. *arXiv preprint arXiv:2307.01850*.

Stavros Anagnou, Christoph Salge, and Peter R Lewis. 2025. Uncertainty, bias and the institution bootstrapping problem. *arXiv preprint arXiv:2504.21579*.

Bowen Baker, Ingmar Kanitscheider, Todor Marber, and 1 others. 2020. Emergent tool use from multi-agent autocurricula. *ICLR*.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, and 1 others. 2018. Emergent complexity via multi-agent competition. *ICLR*.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*.

Richard Bornemann, Gautier Hamon, Eleni Nisioti, and Clément Moulin-Frier. 2023. Emergence of collective open-ended exploration from decentralized meta-reinforcement learning. *NeurIPS Workshop ALOE*.

Marcantonio Bracale Syrnikov, Federico Pierucci, and 1 others. 2026. Institutional AI: Governing LLM collusion in multi-agent cournot markets via public governance graphs. *arXiv preprint arXiv:2601.11369*.

Shaofei Cai and 1 others. 2024a. GROOT: Learning to follow instructions by watching gameplay videos. *arXiv preprint arXiv:2310.08235*.

Tianle Cai, Xuezhi Wang, Tengyu Ma, and 1 others. 2024b. Large language models as tool makers. *ICLR*. ArXiv:2305.17126.

Chen and 1 others. 2025a. AgentScaler: Towards general agentic intelligence via environment scaling. *arXiv preprint arXiv:2509.13311*.

Mengyuan Chen, Chengjun Dai, Xinyang Dong, Chengzhe Feng, Kewei Fu, Jianshe Li, Zhihan Peng, Yongqi Tong, Junshao Zhang, and Hong Zhu. 2025b. Dingtalk DeepResearch: A unified multi agent framework for adaptive intelligence in enterprise environments. *arXiv preprint arXiv:2510.24760*.

Simin Chen, Pranav Pusarla, and Baishakhi Ray. 2025c. Dynamic benchmarking of reasoning capabilities in code LLMs under data contamination. *arXiv preprint arXiv:2503.04149*.

ClawSafety Team. 2026. ClawSafety: "safe" LLMs, unsafe agents. *arXiv preprint arXiv:2604.01438*.

Cui and 1 others. 2025. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*.

Wojciech M Czarnecki, Gauthier Gidel, Brendan Tracey, and 1 others. 2020. Real world games look like spinning tops. *NeurIPS*.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, and 1 others. 2020. Emergent complexity and zero-shot transfer via unsupervised environment design. *NeurIPS*.

Elvis Dohmatob, Yunzhen Feng, Arjun Subramonian, and Julia Ren. 2024. A tale of tails: Model collapse as a change of scaling laws. *arXiv preprint arXiv:2402.07043*.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, and 1 others. 2022. MineDojo: Building open-ended embodied agents with internet-scale knowledge. *NeurIPS*.

Shicheng Fang, Yuxin Wang, Xiaoran Liu, Jiahao Lu, Chuanyuan Tan, Xinchi Chen, Yining Zheng, Xuanjing Huang, and Xipeng Qiu. 2026. AgentLongBench: A controllable long benchmark for long-contexts agents via environment rollouts. *arXiv preprint arXiv:2601.20730*.

Chen Gao and 1 others. 2025. AgentSociety: Large-scale simulation of LLM-driven generative agents. *arXiv preprint arXiv:2502.08691*.

Gu and 1 others. 2025. A survey on trustworthy LLM agents: Threats and countermeasures. *arXiv preprint arXiv:2503.09648*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, and 1 others. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Ziwen Han and 1 others. 2025. Search-time data contamination. *arXiv preprint arXiv:2508.13180*.

Tiantian He, Yihang Chen, Keyue Jiang, and 1 others. 2026. EE-MCP: Self-evolving MCP-GUI agents via automated environment generation. *arXiv preprint arXiv:2604.09815*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *ICLR*. ArXiv:2009.03300.

Josef Hofbauer and Karl Sigmund. 1998. *Evolutionary Games and Population Dynamics*. Cambridge University Press.

John H Holland. 1992. *Adaptation in Natural and Artificial Systems*. MIT Press.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, and 1 others. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. *ICLR*. ArXiv:2308.00352.

Huang and 1 others. 2026a. Emergent social intelligence risks in generative multi-agent systems. *arXiv preprint arXiv:2603.27771*.

Xu Huang, Junwu Chen, Yuxing Fei, and 1 others. 2026b. CASCADE: Cumulative agentic skill creation through autonomous development and evolution. *arXiv preprint arXiv:2512.23880*.

Yuchen Huang, Sijia Li, Minghao Liu, and 1 others. 2025. EnviSAgE: A survey of environment scaling for qualitative agentic experience collection. *arXiv preprint arXiv:2511.09586*.

Haonian Ji and 1 others. 2026. ClawArena: Benchmarking AI agents in evolving information environments. *arXiv preprint arXiv:2604.04202*.

Sihang Jiang, Lipeng Ma, Zhonghua Hong, Keyi Wang, Zhiyu Lu, Shisong Chen, Jinghao Zhang, Tianjun Pan, Weijia Zhou, Jiaqing Liang, and Yanghua Xiao.

2026. SEA-Eval: A benchmark for evaluating self-evolving agents beyond episodic assessment. *arXiv preprint arXiv:2604.08988*.

Carlos E Jimenez, John Yang, Alexander Wettig, and 1 others. 2024. SWE-bench: Can language models resolve real-world GitHub issues? *ICLR*. ArXiv:2310.06770.

Muhammad Ahmad Khan and 1 others. 2026. On the adoption of AI coding agents in open-source Android and iOS development. *arXiv preprint arXiv:2602.12144*.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124. Association for Computational Linguistics.

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

Kumar and 1 others. 2026. Digital red queen: Adversarial program evolution in core war with LLMs. *arXiv preprint arXiv:2601.03335*.

Kevin N Laland, Tobias Uller, Marcus W Feldman, and 1 others. 2015. The extended evolutionary synthesis: Its structure, assumptions and predictions. *Proceedings of the Royal Society B*, 282:20151019.

Hao Li and 1 others. 2026. AIDev: Studying AI coding agents on GitHub. *arXiv preprint arXiv:2602.09185*.

Jing-Jing Li and 1 others. 2025. STAC: When innocent tools form dangerous chains to jailbreak LLM agents. *arXiv preprint arXiv:2509.25624*.

Qi Li and Xinchao Wang. 2026. Sponge tool attack: Stealthy denial-of-efficiency against tool-augmented agentic reasoning. *arXiv preprint arXiv:2601.17566*.

Percy Liang, Rishi Bommasani, Tony Lee, and 1 others. 2023. Holistic evaluation of language models. *Transactions on Machine Learning Research*. ArXiv:2211.09110.

Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. 2023. ToxicChat: Unveiling hidden challenges of toxicity detection in real-world user-AI conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4694–4702. Association for Computational Linguistics.

Liu and 1 others. 2026. Phase transition for budgeted multi-agent synergy. *arXiv preprint arXiv:2601.17311*.

Xiao Liu, Hao Yu, Hanchen Zhang, and 1 others. 2024. AgentBench: Evaluating LLMs as agents. *ICLR*. ArXiv:2308.03688.

Junyu Luo and 1 others. 2025. A survey of self-evolving agents. *arXiv preprint arXiv:2507.21046*.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. AgentBoard: An analytical evaluation board of multi-turn LLM agents. *NeurIPS*. ArXiv:2401.13178.

John Maynard Smith. 1982. *Evolution and the Theory of Games*. Cambridge University Press.

Mead and 1 others. 2026. Improving regret approximation for unsupervised dynamic environment generation. *arXiv preprint arXiv:2601.14957*.

Mieczkowski and 1 others. 2025. Predicting multi-agent specialization via task parallelizability. *arXiv preprint arXiv:2503.15703*.

Melanie Mitchell. 2009. *Complexity: A Guided Tour*. Oxford University Press.

Martin A Nowak. 2006. *Evolutionary Dynamics: Exploring the Equations of Life*. Harvard University Press.

F John Odling-Smee, Kevin N Laland, and Marcus W Feldman. 2003. *Niche Construction: The Neglected Process in Evolution*. Princeton University Press.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, and 1 others. 2021. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*.

Licheng Pan, Yongqi Tong, Xin Zhang, Xiaolu Zhang, Jun Zhou, and Zhixuan Chu. 2025. Understanding and mitigating overrefusal in LLMs from an unveiling perspective of safety decision boundary. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21057–21075. Association for Computational Linguistics.

Licheng Pan, Haochen Yang, Haoxuan Li, Yunsheng Lu, Yongqi Tong, Yinuo Wang, Shijian Wang, Zhixuan Chu, Lei Shen, Yuan Lu, and Hao Wang. 2026. Optimal transport for LLM reward modeling from noisy preference. *arXiv preprint arXiv:2605.06036*.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, and 1 others. 2023. Generative agents: Interactive simulacra of human behavior. *UIST*.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, and 1 others. 2022. Evolving curricula with regret-based environment design. *ICML*.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive APIs. *arXiv preprint arXiv:2305.15334*.

Cheng Qian, Chi Han, Yi R Fung, and 1 others. 2023. CREATOR: Tool creation for disentangling abstract and concrete reasoning of large language models. *EMNLP Findings*.

Yujia Qin, Shihao Liang, Yining Ye, and 1 others. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912. Association for Computational Linguistics.

Mikayel Samvelyan, Akbir Khan, Michael Dennis, and 1 others. 2023. MAESTRO: Open-ended environment design for multi-agent reinforcement learning. *ICLR*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, and 1 others. 2023. Toolformer: Language models can teach themselves to use tools. *NeurIPS*. ArXiv:2302.04761.

Akhil Sharma, Shaikh Yaser Arafat, and 1 others. 2025. XAMT: Bilevel optimization for covert memory tampering in heterogeneous multi-agent architectures. *arXiv preprint arXiv:2512.15790*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, and 1 others. 2023. Reflexion: Language agents with verbal reinforcement learning. *NeurIPS*.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, and 1 others. 2024. The curse of recursion: Training on generated data makes models forget. *Nature*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, and 1 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*. ArXiv:2206.04615.

Kenneth O Stanley, Joel Lehman, and Lisa Soros. 2019. Open-ended search, novelty, and creativity. *Artificial Life*.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*, 2nd edition. MIT Press.

Wang and 1 others. 2026. Agent world model: Infinity synthetic environments for agentic reinforcement learning. *arXiv preprint arXiv:2602.10090*.

Boxuan Wang, Haonan Duan, Yanhao Feng, and 1 others. 2024a. Can LLMs understand social norms in autonomous driving games? *arXiv preprint arXiv:2408.12680*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, and 1 others. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Lei Wang, Chen Ma, Xueyang Feng, and 1 others. 2024b. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

Sizhe Wang, Yongqi Tong, Hengyuan Zhang, Dawei Li, Xin Zhang, and Tianlong Chen. 2025. BPO: Towards balanced preference optimization between knowledge breadth and depth in alignment. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 8811–8826. Association for Computational Linguistics.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, and 1 others. 2024. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *ICLR*. ArXiv:2308.08155.

Xiyang Wu, Zongxia Li, Guangyao Shi, Alexander Duffy, Tyler Marques, Matthew Lyle Olson, Tianyi Zhou, and Dinesh Manocha. 2026. Co-evolving LLM decision and skill bank agents for long-horizon tasks. *arXiv preprint arXiv:2604.20987*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, and 1 others. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Shiyu Xiang, Tong Zhang, and Ronghao Chen. 2025. ALRPHFS: Adversarially learned risk patterns with hierarchical fast & slow reasoning for robust agent defense. *arXiv preprint arXiv:2505.19260*.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*.

Qianqiao Xu, Zhiliang Tian, Hongyan Wu, and 1 others. 2024. Learn to disguise: Avoid refusal responses via multi-agent attacker-disguiser game. *arXiv preprint arXiv:2404.02532*.

Zhihao Xu, Yongqi Tong, Xin Zhang, Jun Zhou, and Xiting Wang. 2025. REWARD CONSISTENCY: Improving multi-objective alignment from a data-centric perspective. *arXiv preprint arXiv:2504.11337*.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Berkeley function-calling leaderboard. `https://gorilla.cs.berkeley.edu/leaderboard.html`.

John Yang, Carlos E Jimenez, Alexander Wettig, and 1 others. 2024. SWE-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*.

Shidong Yang, Ziyu Ma, Tongwen Huang, Yiming Hu, Yong Wang, and Xiangxiang Chu. 2026. CoEvolve: Training LLM agents via agent-data mutual evolution. *arXiv preprint arXiv:2604.15840*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, and 1 others. 2023. ReAct: Synergizing reasoning and acting in language models. *ICLR*.

Bowen Ye and 1 others. 2026. Claw-Eval: Towards trustworthy evaluation of autonomous agents. *arXiv preprint arXiv:2604.06132*.

Yuan and 1 others. 2024. A comprehensive survey in LLM(-agent) full stack safety. *arXiv preprint arXiv:2504.15585*.

Zhang and 1 others. 2026a. A systematic survey of self-evolving agents: From model-centric to environment-driven co-evolution. *SSRN preprint*.

Qifan Zhang, Dongyang Ma, Tianqing Fang, Jia Li, Jing Tang, Nuo Chen, Haitao Mi, and Yan Wang. 2026b. Training LLM agents for spontaneous, reward-free self-evolution via world knowledge exploration. *arXiv preprint arXiv:2604.18131*.

Shuyan Zhou, Frank F Xu, Hao Zhu, and 1 others. 2024. WebArena: A realistic web environment for building autonomous agents. *ICLR*. ArXiv:2307.13854.

A Scope and Classification Details

The corpus is organized around inherited-environment mechanisms. We began from seed papers in agent benchmarks, tool and memory agents, self-play, environment design, model collapse, software agents, multi-agent systems, and safety, then expanded by citation following, adjacent-survey comparison, and targeted keyword search. The goal is mechanism coverage rather than prevalence estimation.

We include a work when it satisfies at least one criterion: it studies an agent action that persistently changes the environment; it studies environmental structure as a driver of capability; or it studies a coupled adaptive process involving feedback, competition, accumulation, or degradation. We exclude one-step prompts or inputs, purely episodic tool or memory use, architecture surveys without an inheritable environment variable, and results not tied to persistence, feedback, or evaluation. Borderline cases are included only when persistence changes the later evaluation environment; ordinary scratchpads, retrieval, and discarded one-off tool calls are excluded even if they improve the current episode.

Classification fields and limits. Each included work is assigned one primary feedback direction, inherited variable, persistence mechanism, and regime. Secondary tags are retained only when they affect the evaluation argument. Claims about benchmark results, dataset sizes, and system behavior are taken from primary papers or official benchmark documentation when available. The supplementary material reports row-level corpus coding, search procedure summaries, and a 24-row stratified consistency audit. This makes the mechanism coding and search trail inspectable, but it is not a PRISMA-style systematic review and does not provide exhaustive exclusion counts or inter-annotator agreement.

B Compact Background Lenses and Regime Map

This appendix section keeps only the background needed to audit the survey taxonomy. The theories below are interpretive lenses for inherited-state evaluation, not independent claims that the paper proves across all agent systems. They explain why the corpus codes persistence, feedback direction, reversibility, governance, and degradation, while the main contribution remains the Artifact Ecology measurement protocol.

Background lenses. Complex adaptive systems motivate population-level variables such as conventions, institutions, and market structure (Holland, 1992; Mitchell, 2009). Niche construction motivates treating agent-created tools, memories, repositories, and datasets as ecological inheritance (Odling-Smee et al., 2003; Laland et al., 2015). Evolutionary game theory motivates non-transitivity, cycling, and arms-race diagnostics in competitive or strategic settings (Maynard Smith, 1982; Nowak, 2006; Czarnecki et al., 2020). Open-ended evolution motivates the need to preserve variation and useful innovations when environments continually generate new challenges (Stanley et al., 2019; Open Ended Learning Team et al., 2021; Samvelyan et al., 2023).

Survey line	Primary organizing variable	Coverage relevant to inheritance	Gap addressed here
LLM-agent surveys (Wang et al., 2024b; Xi et al., 2023; Guo et al., 2024)	Agent architectures and capabilities.	Memory, tool use, planning, and multi-agent workflows.	They rarely compare reset conditions with environments that preserve and expose inherited state.
Environment-scaling surveys (Huang et al., 2025)	Environment generation and capability scaling.	Task distributions, generated environments, and feedback loops.	They emphasize environments as capability sources, not agent-written artifacts as downstream evaluation objects.
Self-evolving-agent surveys (Luo et al., 2025; Zhang et al., 2026a)	Iterative self-improvement and adaptation.	Self-generated data, tools, memory, and environment updates.	They do not center artifact usefulness, debt, governance, and transfer under reset-vs-inheritance protocols.
Safety and trust surveys (Yuan et al., 2024; Gu et al., 2025)	Risks, defenses, and deployment trust.	Tool risk, memory risk, oversight, and policy mechanisms.	They treat safety mechanisms as defenses, not as inherited policy artifacts to be admitted, audited, deprecated, or reused.

Table 3: Positioning relative to adjacent surveys. This paper uses inherited environment state as the primary organizing variable.

Evaluation family	What persists	Reuse/exposure accounting	Governance state	Held-out inheritance	Remaining gap addressed by Artifact Ecology
Dynamic data and adversarial evaluation (Ribeiro et al., 2020; Kiela et al., 2021)	Test items or data collection process may change.	Usually no agent-written artifact library.	Usually no.	Usually no.	Does not ask whether agent outputs become future tools, memory, policies, or workspace state.
Interactive agent benchmarks (Liu et al., 2024; Zhou et al., 2024; Xie et al., 2024; Drouin et al., 2024)	Environment state exists inside an episode; tasks are commonly reset for comparability.	Limited or task-local.	Usually no.	Usually no.	Downstream state across benchmark episodes is erased or treated as contamination.
Software-agent benchmarks (Jimenez et al., 2024; Yang et al., 2024)	Repository patches, tests, and logs are task outputs.	Patch content can be inspected, but cross-task artifact reuse is not the score object.	Usually outside the benchmark.	Usually no.	Does not compare reset repositories with inherited code/test/tool ecologies across task streams.
Tool and memory evaluation (Schick et al., 2023; Qin et al., 2024; Yan et al., 2024; Shinn et al., 2023)	Tools or memory may be available to the agent.	Often measures use of provided tools or within-run memory.	Rarely.	Rarely.	Does not jointly score creation, later exposure, reuse, rejection, deprecation, and transfer of agent-created artifacts.
Mutable or long-horizon agent evaluation (Chen et al., 2025c; Ji et al., 2026; Fang et al., 2026; Jiang et al., 2026)	Tasks, information environments, or long rollouts can change.	Partial trajectory accounting.	Partial or domain-specific.	Partial.	Needs a common artifact-level report for visibility, reuse, governance, transfer, error propagation, and cost.
ARTIFACT ECOLOGY wrapper	Tools, memory, code, documents, schemas, caches, policies, and audit logs.	Explicit artifact visibility and reuse fields.	Admission, rejection, deprecation, and audit records.	Clean versus inherited held-out tasks.	Adds a protocol layer over existing tasks and verifiers so inherited state becomes a first-class evaluation object.

Table 4: Detailed comparison with closest evaluation families. The proposed contribution is not that each column is new in isolation, but that a benchmark condition should jointly preserve, expose, govern, transfer-test, and score inherited agent-created artifacts.

Boundary for social and institutional state. Multi-agent studies report routines, norms, collusion, traffic conventions, and institutional effects (Park et al., 2023; Gao et al., 2025; Wang et al., 2024a; Bracale Syrnikov et al., 2026; Anagnou et al., 2025). For Artifact Ecology, however, latent norms count only when they are represented through inspectable logs, roles, protocols, policies, sanctions, governance records, or other artifacts that later agents can inherit and auditors can inspect. This keeps broad inherited environment state separate from the narrower auditable protocol layer.

Safety and governance. Safety is treated as coupled inherited state rather than as a static model property. Agent actions can change tool access, memory, permissions, policies, and threat surfaces; attackers, defenders, and governance environments may then adapt to one another (ClawSafety Team, 2026; Li et al., 2025; Li and Wang, 2026; Sharma et al., 2025; Xu et al., 2024). The point is not to force every defense into Definition 1's f operator, but to record which safety-relevant artifacts and policies are admitted, rejected, deprecated, or exposed downstream.

Compact regime map. The regimes in Table 7 are diagnostic patterns rather than deterministic laws. A stream can move between regimes as horizon length, governance strength, artifact diversity, and context cost change. Collapse-like outcomes include shared-state degradation, conformity, contamination, or degraded collective outcomes; they need not imply a literal fixed point.

C Extended Taxonomy Overview

Figure 2 gives the one-page mechanism map used to organize the 133-reference corpus. The map is an inherited-state and feedback taxonomy rather than a second survey narrative.

D Task-Pool Curation Summary

The probe task-selection pipeline uses deterministic metadata extraction followed by task-level curation. The materialized pool contains 3,254 tasks from Claw-Bench/Claw-Eval (Ye et al., 2026),

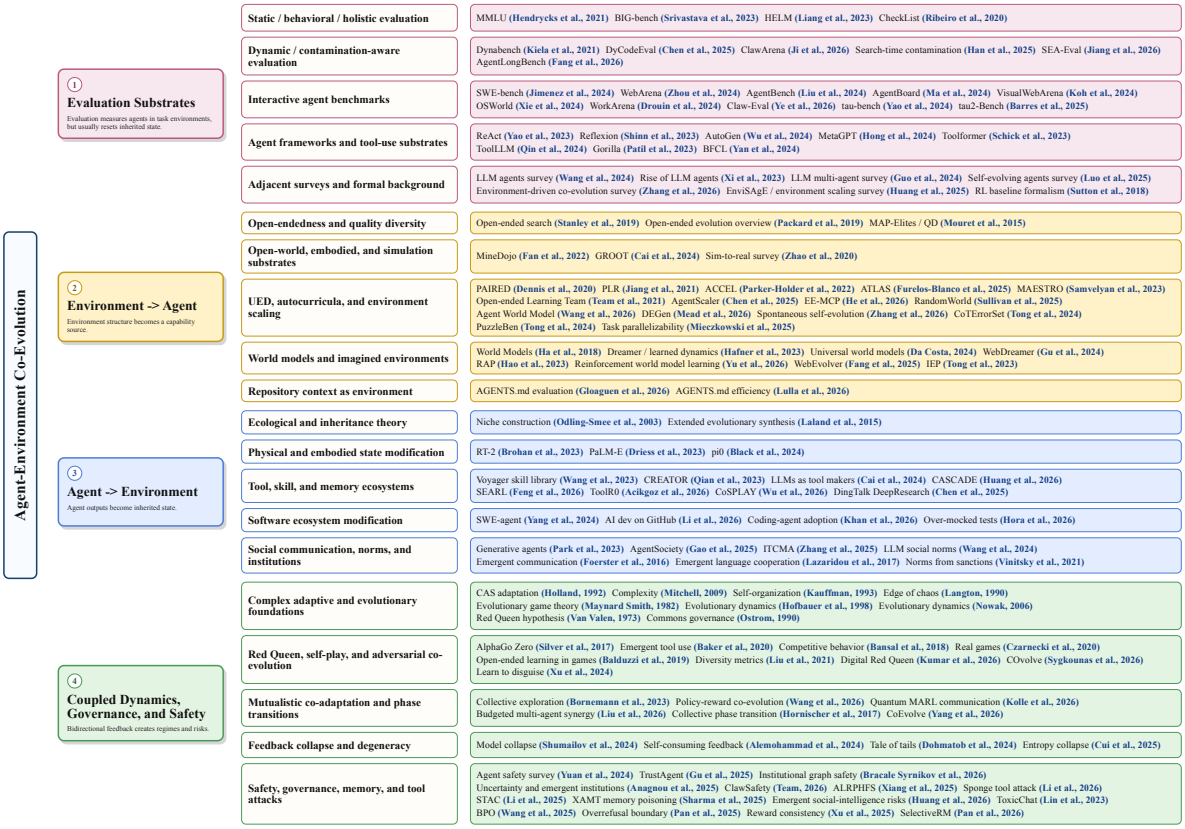

Figure 2: Full mechanism map of the 133-reference corpus. The map organizes selected work into evaluation substrates, environment-to-agent capability sources, agent-to-environment persistence mechanisms, and coupled dynamics involving governance and safety.

Corpus step	Criterion	Output	Role in the survey
Seed selection	Areas where agent actions or environment structure persist across episodes or generations.	Initial papers from agent benchmarks, tool/memory agents, environment design, software agents, model collapse, multi-agent systems, and safety.	Ensures coverage of both environment→agent and agent→environment mechanisms.
Expansion	Backward/forward citation following, adjacent-survey comparison, and targeted keyword search.	Additional work connected to inheritance, persistence, feedback, adaptation, or governance.	Reduces dependence on any single community's terminology.
Inclusion/exclusion	Include persistent feedback paths; exclude one-step prompts, purely episodic tool use, and architecture surveys without inheritable state.	133-reference corpus.	Keeps the survey focused on evaluation-relevant inheritance.
Coding	Assign feedback direction, inherited variable, persistence mechanism, and regime.	Taxonomy used in Figure 2.	Converts a cross-domain corpus into evaluation categories and metrics.
Row-level coding	Record citation key, direction, variable, mechanism, regime, rationale, and tags for each included work.	133 coded entries.	Lets readers audit how individual citations support the taxonomy.
Search audit	Record query families, source families, snapshot date, and screening rule.	Query-family expansion summary.	Lets readers audit the expansion procedure while preserving the paper's mechanism-driven scope.
Coding validation	Re-check a stratified sample of ledger rows across theory, evaluation, software, data, safety, and tool/memory strata.	24-row second-pass consistency audit.	Tests whether the primary coding decisions survive a compact consistency check.

Table 5: Corpus construction and coding schema. The survey is organized around mechanisms of inherited environment state rather than around a single benchmark family, agent architecture, or application domain.

Lens	Inheritance relevance	Diagnostic use here	Representative evidence
Complex adaptive systems	Local interactions can produce macro-level routines, norms, institutions, or market structure.	Look beyond single-agent scores to shared state and population-level traces.	Smallville routines (Park et al., 2023); Cournot-market collusion/governance (Bracale Syrnikov et al., 2026).
Niche construction	Agents can modify the environment later agents inherit.	Treat tools, code, data traces, memories, and policies as evaluation objects when persistent and auditable.	Tool inheritance (Wang et al., 2023; Huang et al., 2026b); model-collapse feedback (Shumailov et al., 2024).
Evolutionary game theory	Payoffs can depend on other agents' strategies, producing cycling or arms races.	Track strategic diversity, counter-adaptation, and governance pressure rather than endpoint score alone.	Non-transitive dynamics (Czarnecki et al., 2020); hide-and-seek arms race (Baker et al., 2020).
Open-ended evolution	Novelty requires variation, preservation, and continuing challenge.	Separate useful accumulation from saturation, context cost, and brittle dependence on inherited artifacts.	Dynamic curricula and open-ended systems (Open Ended Learning Team et al., 2021; Parker-Holder et al., 2022; Bornemann et al., 2023).

Table 6: Compact theoretical lenses used only to motivate the inherited-state coding axes. They are background for the survey taxonomy, not a separate theoretical survey.

τ-bench (Yao et al., 2024), τ^2-bench (Barres et al., 2025), Gorilla/BFCL (Patil et al., 2023; Yan et al., 2024), and SWE-bench Verified (Jimenez et al., 2024). This curation is only for executable probe construction; it is not used for deciding survey-corpus inclusion. The criterion is inherited-environment signal rather than task difficulty: strong candidates have repeated related episodes, domain state or tool/schema structure, artifacts that can help or harm later episodes, and measurable outcomes; weak candidates are isolated one-shot calls where any inherited artifact would be superficial.

E Executable Probe Details

Conditions. The executable probes are measurement checks, not model rankings. Each comparison fixes the task source, task order, model/backend, decoding setting, interaction budget, native verifier, and workspace construction. The comparison factor is the inheritance wrapper: whether prior artifacts are exposed, whether candidate artifacts are admitted by governance, or whether a fixed inherited library is preseeded at a controlled size. This is diagnostic rather than an isolated causal estimand.

Logging. The main Claw-Bench stream contains 95 selected tasks, and the BFCL probe uses a 20-task multi-turn function-calling slice. The wrapper represents artifacts as typed shared state, includ-

Regime	Feedback pattern	Trigger or structural condition	Diagnostic clue	Empirical examples
Red Queen / arms race	Counter-adaptive or negative feedback.	Competing agents, attacks, or environments adapt on similar timescales.	Escalation, cycling, or constant effort to maintain relative performance.	Self-play and hide-and-seek (Baker et al., 2020); Digital Red Queen (Kumar et al., 2026).
Mutualistic accumulation	Bounded positive feedback.	Artifacts or conventions are useful, recoverable, and governed enough to remain productive.	Increasing reuse or capability without runaway cost, brittle dependence, or unchecked propagation.	Voyager skills (Wang et al., 2023); cumulative skill creation (Huang et al., 2026b).
Threshold shift / phase transition	Mixed or nonlinear feedback.	Additional agents, artifacts, context, or communication quality crosses a threshold.	Abrupt change in behavior, cooperation, collapse, or score not predicted by smooth trends.	Strategy shifts (Baker et al., 2020); budgeted synergy/collapse (Liu et al., 2026).
Collapse-like degradation	Unbounded positive feedback with weak correction.	Hard-to-reverse contamination, low diversity, or unmanaged shared-state dependence.	Distributional narrowing, conformity, error amplification, or degraded shared outcomes.	Model collapse (Shumailov et al., 2024); entropy collapse (Cui et al., 2025); strategic degradation (Huang et al., 2026a).

Table 7: Diagnostic regime map for inherited environment state. The columns are typical patterns and clues, not necessary-and-sufficient conditions.

ing tools, tests, documentation, and memory. Each episode records condition, task identity, verifier outcome, artifact state before and after the episode, governance decisions, explicit reuse evidence, runtime/cost metadata, and transcripts. BFCL records a raw response field, `used_inherited_artifact`, which is normalized into the episode-level trace label `artifact_reused`; Claw-Bench records `artifact_diff.reused` and checks it against transcript artifact-name evidence.

Validation limits. Reuse is counted only when a trajectory explicitly cites an inherited note, calls an inherited tool, imports an inherited helper, or bases a path/schema/tool choice on inherited artifact content. This conservative definition is auditable but misses implicit influence. The library-size ablation uses the same 24-task slice at all six settings and fixes model/backend, task order, native verifier, interaction budget, and no-writer policy; seeded artifacts are nested prefixes from a fixed ordered list of non-poisoned prior artifacts. The reproducibility unit is therefore larger than a score table: it includes task specifications, run configuration, traces, transcripts, verifier outcomes, artifact snapshots, governance logs, and summary code. These checks make labels inspectable, but they are not substitutes for repeated seeds, shuffled task orders, matched-context resets, artifact ablations, negative Claw-Bench reuse sampling, or a full independent annotation study.

Artifact visibility and reuse. Visibility is counted when the wrapper makes an inherited artifact available in the agent's workspace or prompt

Component	Audit role
Task specification	Task ids, source benchmark, order, split, and selection metadata; separates inheritance effects from task-selection and order effects.
Run configuration	Model/backend, decoding, budget, condition, and policy; makes reset, persistent, and governed runs comparable.
Episode evidence	Per-task result, costs, artifact states, transcripts, explicit reuse evidence, and verifier outputs.
Artifact state	Snapshots plus admission, rejection, deprecation, and test logs; makes inheritance and governance inspectable.
Aggregation	Scripts for scores, visibility, reuse, costs, artifact counts, and sampled validation checks.

Table 8: Compact audit bundle for reproducible Artifact Ecology evaluation.

Condition	Diagnostic contrast
`static_reset`	Clean environment per task; standard isolated task capability.
`persistent`	Later tasks inherit all prior artifacts; tests ungoverned accumulation.
`persistent_governed`	Later tasks inherit only admitted artifacts; tests admission rules.
`inheritance_eval`	Held-out tasks run in clean versus inherited environments.

Table 9: Artifact Ecology protocol conditions; reset remains the isolated-competence anchor.

context. Reuse is counted only when the trajectory explicitly cites an inherited note, calls an inherited tool, imports an inherited helper, or bases a path/schema/tool choice on inherited artifact content. This conservative definition undercounts implicit influence, but avoids treating mere availability as usefulness.

Governance. Candidate Python tools must have a matching test file, compile successfully, and pass a smoke test before admission. Candidate memory or documentation artifacts must state their task source and intended reuse scope. Rejected artifacts remain logged for audit but are not exposed to later episodes. The governed condition therefore measures both task score and the administrative trace created by artifact admission.

Condition	Score	Logged	Admitted	Rejected
Static reset	9/95	85	N/A	N/A
Persistent	16/95	102	N/A	N/A
Governed	13/95	118	7	27

Table 10: Governance results for the Claw-Bench diagnostic stream. Admission and rejection are defined only for the governed condition. Of 118 governed candidates logged, 7 were admitted and 27 rejected; the rest were filtered before review or were not governance-eligible.

Check	Outcome and limit
Corpus coding audit	24 stratified rows retained their primary direction, inherited variable, mechanism, and regime assignments; this is not full inter-annotator agreement over all 133 works.
BFCL reuse labels	13 trajectories: raw `used_inherited_artifact` responses agreed with normalized `artifact_reused` trace labels; implicit influence is not measured.
Claw-Bench reuse labels	11 positive trajectories: traces listed reused artifacts; transcript scans found five artifact basenames and six weak package-marker matches. Positive reuse need not imply usefulness.

Table 11: Compact validation checks for corpus coding and reuse-label extraction.

Library-size setting	Pass	Pass rate	Reuse	Mean input tokens
0	9/24	37.5%	0.0%	19.6k
5	6/24	25.0%	0.0%	22.2k
10	8/24	33.3%	8.3%	21.9k
25	9/24	37.5%	0.0%	24.1k
50	8/24	33.3%	0.0%	23.2k
100 target (66 materialized)	9/24	37.5%	8.3%	26.7k

Table 12: Controlled artifact-library ablation on a fixed 24-task slice. The comparison factor is the inherited-library target size; the result is intentionally non-monotonic.

Auxiliary probe setup. The 95-task Claw-Bench stream is selected from domains with repeated structure, writable file-system state, reusable schemas, or document/code artifacts. The BFCL probe uses a small function-call slice to test whether the same wrapper can expose inherited path notes in a stateless tool-use benchmark. The held-out Claw probe uses the final two document-editing tasks only as a sanity check for transfer; it is intentionally reported as a negative diagnostic rather than a representative transfer benchmark.

The auxiliary probes stress specific failure modes rather than adding new leaderboards. Bad inheritance seeds poisoned tool or memory artifacts and audits whether they are explicitly reused. Memory ecology varies append, summarize, govern, and deprecate policies on the same small task family. Library size fixes the 24-task slice and varies only the initial inherited artifact count.

Probe	Variable and measurement
Bad inheritance (96)	Poisoned tool/memory artifacts; no detected poisoned-artifact reuse.
Memory ecology (75)	Five memory policies over the same 15-task family; each policy scores 0/15.
Library size (144)	Initial inherited artifact count; 0, 5, 10, 25, 50, and 100 artifacts yield non-monotonic outcomes.

Table 13: Auxiliary diagnostic probes after the main Artifact Ecology stream.

Interpretation. Persistent conditions may provide additional context, so the reported pass-rate differences should not be read as pure estimates of model competence. The central measurement signal is the appearance of variables that reset evaluation cannot express: artifact visibility, explicit reuse, admission/rejection, library growth, failed transfer, and dependence on seeded library size.

F Use of LLMs

We used AI agents to assist with literature search, paper triage, and preliminary classification of works by feedback direction, inherited variable, persistence mechanism, and regime. The authors reviewed and edited the resulting corpus coding and are responsible for the paper's claims and wording.