Evaluating Skills, Not Just Agents: Agentic Continuous Evaluation of Skills (ACES)

Christopher Kevin
christopherk@nvidia.com
NVIDIA
Santa Clara, USA

Roshni Malani
rmalani@nvidia.com
NVIDIA
Santa Clara, USA

Meghana Puvvadi
mpuvvadi@nvidia.com
NVIDIA
Santa Clara, USA

Rama Akkiraju
rakkiraju@nvidia.com
NVIDIA
Santa Clara, USA

Jean-Francois Puget
jpuget@nvidia.com
NVIDIA
Courbevoie, France

Narendran Raghavan
nraghavan@nvidia.com
NVIDIA
Santa Clara, USA

Mohit Gupta
mohgupta@nvidia.com
NVIDIA
Pune, India

Seong Hee Lee
seongheel@nvidia.com
NVIDIA
Santa Clara, USA

Abstract

Agent skills extend LLM agents through natural-language instructions, scripts, examples, and references that agents load on demand. Current repository review gates often scan these artifacts with structural checks, LLM-as-Judge rubrics, script linting, and security rules. Such gates are useful, but they do not answer the runtime question: does a live agent discover the skill, use it correctly, and perform better than it would without that skill?

We present ACES (Agentic Continuous Evaluation of Skills), a repository-native, trajectory-aware framework for evaluating skills as executable agent artifacts. ACES runs paired live trials for each task: a with-skill condition and a matched baseline in which the target skill is withheld while configured support and decoy skills remain fixed. The resulting trajectories are normalized into the Agent Trajectory Interchange Format (ATIF) and graded by six default metrics: security, skill execution, skill efficiency, accuracy, goal accuracy, and behavior check. Their paired deltas yield *Skill Lift*, an estimate of the target skill's added value for a fixed task, harness, workspace, and grading policy.

On 145 real skills from internal enterprise repositories and public skill catalogs, scan-only gates surface real authoring issues but measure complementary facets: structural and LLM-judge scores correlate at Spearman $\rho = 0.14$. Across the live-trial surface, ACES exercised 89 unique skill variants. For the main production-skill analysis we exclude scaling stress variants and report 947 paired task cases from 64 production skills and four primary harnesses. Mean overall Skill Lift is 0.2134 (95% paired-case CI [0.1967, 0.2301]), with positive lift in 72.8% of paired cases; the largest mean gains appear in skill execution, behavior check, and skill efficiency, which are runtime-process signals that document scans cannot observe. ACES also makes evaluation assets first-class: author-owned datasets, trajectory-grounded refinement, `expected_behavior` checks, BYOT tasks, BYOG graders, and CI reporting. We additionally describe a product-first operating mode that applies the same paired protocol to bundles, teams, and plugins while reserving Skill Lift for skill-centered comparisons.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; *Natural language processing*; • **Software and its engineering** → *Software verification and validation.*

Keywords: agent skills, LLM agents, evaluation, LLM-as-Judge, Skill Lift, multi-agent evaluation, continuous evaluation, CI/CD, procedural knowledge

ACM Reference Format:
Christopher Kevin, Roshni Malani, Meghana Puvvadi, Rama Akkiraju, Jean-Francois Puget, Narendran Raghavan, Mohit Gupta, and Seong Hee Lee. 2026. Evaluating Skills, Not Just Agents: Agentic Continuous Evaluation of Skills (ACES). In *Agent Skills '26: The First Workshop on Agent Skills, May 26, 2026, San Jose, CA, USA*. ACM, New York, NY, USA, 12 pages.

1 Introduction

Skills are the application layer for agents. Over the past year, LLM-based agents have acquired a repeatable extension mechanism: *agent skills*. A skill is a natural-language package of procedural know-how—a `SKILL.md` description, optional deterministic scripts the agent invokes for steps it should not improvise, plus references and examples—that the agent loads at inference time. The defining design choice is progressive disclosure: the agent reads only the short description up front, and pulls in instructions, scripts, or examples only when it decides the skill applies, so dozens of skills can coexist in a workspace without bloating context [1, 2, 24, 27].

Agent Skills '26, San Jose, CA, USA
2026.

The format is adopted across Claude Code, Codex, Cursor, and other agent harnesses, and community registries already host thousands of user-contributed skills. Bundled sets of related skills are increasingly called plugins; the methodology in this paper applies to a single skill, a plugin, or any composition of skills an agent can load.

The current state of practice. Repository review workflows for skills commonly cluster into four scan-only classes. (1) *Structural checks* enforce a skill specification—frontmatter fields, section layout, script presence, naming conventions. Vendor authoring tools such as Anthropic's skill-creator illustrate this structural-contract workflow [4]. (2) *LLM-as-Judge rubrics* grade subjective qualities of the documentation that static rules cannot see: clarity of instructions, scope definition, example quality, trigger phrasing [9, 16, 19]. (3) *Linters* check the scripts themselves for syntax, style, and dangerous patterns. (4) *Security scanners* detect prompt-injection markers, leaked secrets, destructive shell patterns, and suspicious URL references inside the skill content. All four classes share a structural property: they *scan the skill document* rather than executing paired live trials.

Doc-scanning is necessary, not sufficient. Running a skill through scan-only evaluation is analogous to compiling a program with `-Wall -Werror`: the absence of warnings does not mean the program does what it should. A skill can scan cleanly and still fail at runtime because (a) the agent never discovers it when the user asks a relevant question, (b) the agent reads the documentation but invokes the wrong script or wrong arguments, (c) the agent produces correct output but misinterprets and misreports it, (d) the skill collides with another skill already in the workspace, or (e) the skill is silently regressed after a model update changes how the agent reasons about its documentation, a failure mode observable through longitudinal paired runs. None of these failure modes is observable from the skill alone.

Measuring the gap. We evaluated 145 real skills from internal enterprise repositories and public skill catalogs with both structural and LLM-judge tiers. 94.5% of skills pass the default C-grade gate on structural checks, and 86.2% pass the LLM-judge rubric. And yet the two scores correlate at Spearman $\rho = 0.14$. In other words, even among the scan methods *the two measure different facets*. This is not a claim that either scan is wrong: structural checks capture authoring contract compliance, while LLM rubrics capture clarity and workflow quality. The point is that neither method tells us what the agent actually does with the skill at runtime.

ACES: extending static skill-scanning with live-agent evaluation. We present ACES (Agentic Continuous Evaluation of Skills), a methodology and system for evaluating skills as executable agent artifacts rather than static documents alone. ACES is organized around three portable interfaces: an evaluation-asset contract authored with the skill, an adapter that materializes those assets into paired runtime tasks, and an ATIF-based trajectory contract that lets the same grading layer compare behavior across agent harnesses. For each author-provided task, ACES runs paired conditions: a with-skill condition, where the target skill is available, and a baseline condition, where the target skill is withheld while configured prerequisite, helper, reference, or decoy skills remain fixed. The paired difference yields *Skill Lift*: the added value contributed by the target skill under a fixed agent, model, task, workspace, and grading policy. Our contributions:

1. A methodology for skill-as-artifact evaluation that extends doc-scanning with live-agent evaluation on author-owned tasks, including paired with-skill/baseline measurement and Skill Lift (§2, §3).
2. An evaluation-asset authoring workflow in which authors can write datasets directly, seed or refine cases with LLM assistance and real trajectories, use free-form `expected_behavior` assertions for LLM-judged workflow checks, and attach BYOT/BYOG assets when generic metrics are insufficient (§3.1).
3. A shared ATIF-based grading layer that applies deterministic, security, LLM-judge, and optional domain-specific metrics to trajectories across agent harnesses, with stakeholder-facing dimensions for author review (§3).
4. A dynamic ACES adapter that stages fresh paired task environments across agents, task sources, workspace modes, grading modes, and sandbox backends, including isolated and group workspaces for skill-selection and prerequisite-skill scenarios (§3).
5. An evaluation-native skill-development workflow in which structural checks, LLM-judge scoring, security scanning, optional live-agent evaluation, and Skill Lift reports become review evidence for skill changes across product repositories, skill-centric repositories, and registries (§3).
6. An empirical study on 145 real skills from internal enterprise repositories and public skill catalogs showing that scan-only signals are incomplete: 94.5% pass the default structural gate despite widespread frontmatter violations, and deterministic and LLM-judge scores correlate weakly at Spearman $\rho = 0.14$. The live trials cover 89 unique skill variants; the main production-skill subset has 64 skills and 947 paired task cases across four primary harnesses, with positive mean Skill Lift (0.2134, 95% CI [0.1967, 0.2301]) and the largest mean gains in execution, behavior check, and skill efficiency. The harness/model slice shows why paired measurement matters: each lift value is interpreted against its own matched baseline rather than as an absolute model ranking (§4).

We defer related work to §6 so the methodology and measured results land first.

2 Evaluating the Skill Artifact

Repository review workflows for skills usually begin with four scan-only gates. Structural checks enforce frontmatter, naming, layout, script presence, and token-budget rules; LLM-as-Judge rubrics score clarity, examples, scope, trigger language, and workflow completeness [9, 19]; script linting catches syntax and style failures; and security scanners look for prompt injection, leaked secrets, destructive shell patterns, and suspicious URLs. These gates are cheap and useful. They are also artifact-only: they do not observe whether a live agent discovers the skill, follows its workflow, invokes the right tool, or avoids a confusing neighbor skill.

On 145 real skills drawn from internal enterprise repositories and public skill catalogs, the four doc-scanning classes surface concrete issues. Structural scores range from 61 to 98 (mean 79.2, median 79.8, $\sigma = 4.9$). **94.5%** of skills pass the default 70-point gate on first submission, but only **48.9%** clear 80 points—a visible C-to-B wall (Figure 1, left). Per-repository mean scores are similar in the 78.5–80.2 range, while skill-to-skill variation remains visible ($\sigma = 4.9$). The point of Figure 1 is not that most skills are runtime-ready; it is that a permissive structural gate is good for surfacing authoring issues but too weak to stand in for live skill behavior.

The scans measure different facets. This is the empirical fulcrum of the paper. On the same 145 skills, the deterministic structural score and the LLM-judge overall score correlate at **Spearman** $\rho = 0.14$ (Pearson $r = 0.08$; Figure 1, right). Both scales pass a high fraction of skills individually (94.5% and 86.2% respectively at threshold 70), so the near-zero correlation is not an artifact of one method being degenerate. The difference clusters in two patterns: skills that pass structure but fail the rubric have correct fields with vague or ambiguous instructions; skills that fail structure but pass the rubric are content-rich but drop declared metadata. The two scan methods therefore measure different facets of skill quality and *they do not collapse to the same signal.* That complementarity is useful in review, but it also means a single scan score should not be treated as runtime evidence. The purpose of Figure 1 is therefore diagnostic: it shows two complementary scan-only views before we ever ask whether the skill helps a live agent.

The most common deterministic findings are frontmatter and documentation omissions: 99.3% of skills fail to declare `tools`, 97.9% omit `Limitations`, 97.2% omit `author`, and 91.7% omit `tags`. These are useful authoring signals, but the high pass rate shows that default scan gates are permissive. Absolute LLM-judge values also depend on the judge: across the nine rubric criteria, the mean criterion-level spread among the three judges is 1.27 points on the 0–10 scale (maximum 2.17), so we treat judge-labeled scores as review aids rather than runtime proof.

The missing observation is runtime behavior. Scan gates cannot tell whether the agent will discover the skill, invoke the right script with the right arguments, interpret outputs correctly, or route among sibling skills. Those questions require a paired live run.

Positioning. Prior skill and agent benchmarks show that skills can affect agent performance [17, 20], and vendor tools help authors inspect or improve skill artifacts [4]. ACES connects these pieces in a repository-native workflow: scan the artifact, run paired live trials across harnesses, normalize trajectories, measure added skill value, and feed review evidence back to authors.

3 Live Agent Evaluation

A live-agent evaluation closes the gap by treating the skill as an executable intervention. ACES follows three design principles: authors write one evaluation contract per skill; runtime value is measured differentially with paired with-skill and baseline runs; and developer intent in `EVAL.md`, BYOT tasks, BYOG graders, and `expected_behavior` assertions outranks generated defaults. In each paired run, the prompt, agent, model, task assets, sandbox, supporting skills, and grader are held fixed while only the target skill's availability changes. Supporting prerequisite, helper, reference, or decoy skills remain in both conditions, which isolates the target skill's added value and preserves realistic routing pressure. Figure 2 summarizes the resulting architecture.

3.1 Evaluation Assets are First-Class Artifacts

The evaluation assets drive the entire live-agent evaluation. In the common path this is an `evals.json` dataset; richer cases may add input fixtures, environment configuration, BYOT task definitions, or BYOG graders. Their quality directly shapes the resulting scores: if the questions do not cover realistic uses, or the `expected_behavior` steps are vague, every evaluator downstream loses signal. We therefore treat evaluation assets as first-class authoring artifacts with their own lifecycle (Figure 3). Operationally, the standard practice we advocate is to create or update `evals/evals.json` while building the skill itself; a skill may be scanned without evaluation assets, but it has not yet declared the runtime behavior that should be preserved.

Evaluation contract. Each dataset entry contains an `id`, user question, `expected_skill`, optional `expected_script`, `ground_truth`, and ordered `expected_behavior` assertions. Behaviors are free-form natural-language checks such as "read `SKILL.md` before executing"; a judge answers yes/no against the trajectory and the score is the fraction satisfied. Authors can write cases directly, bootstrap explicit, implicit, contextual, or negative cases from `SKILL.md` [5, 24], refine them from saved agent trajectories, or override generated

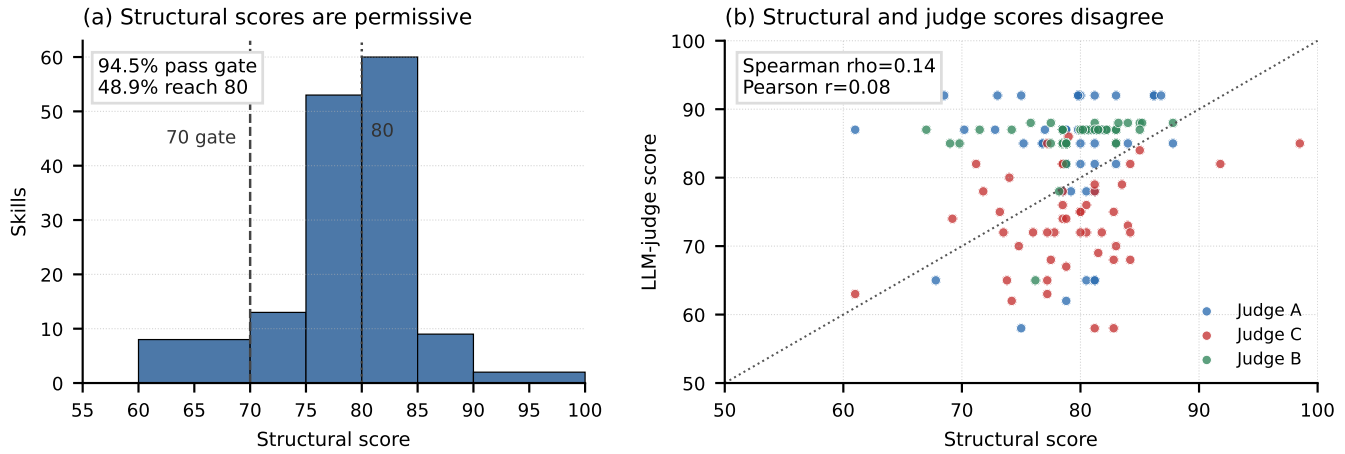

Figure 1. Doc-scan evidence motivates, but does not replace, live evaluation. (a) The 0–100 structural score aggregates roughly 50 deterministic rule checks across Correctness, Discoverability, Reliability, and Efficiency for n=145 real skills. The default 70-point gate is permissive: 94.5% of skills clear it, but only 48.9% reach the 80-point line. (b) Structural scores and LLM-judge rubric scores differ on the same corpus (Spearman $\rho = 0.14$, Pearson $r = 0.08$). Together, the panels show that document scanning surfaces real authoring issues, but neither scan-only axis observes discovery, tool use, workflow order, or task success.

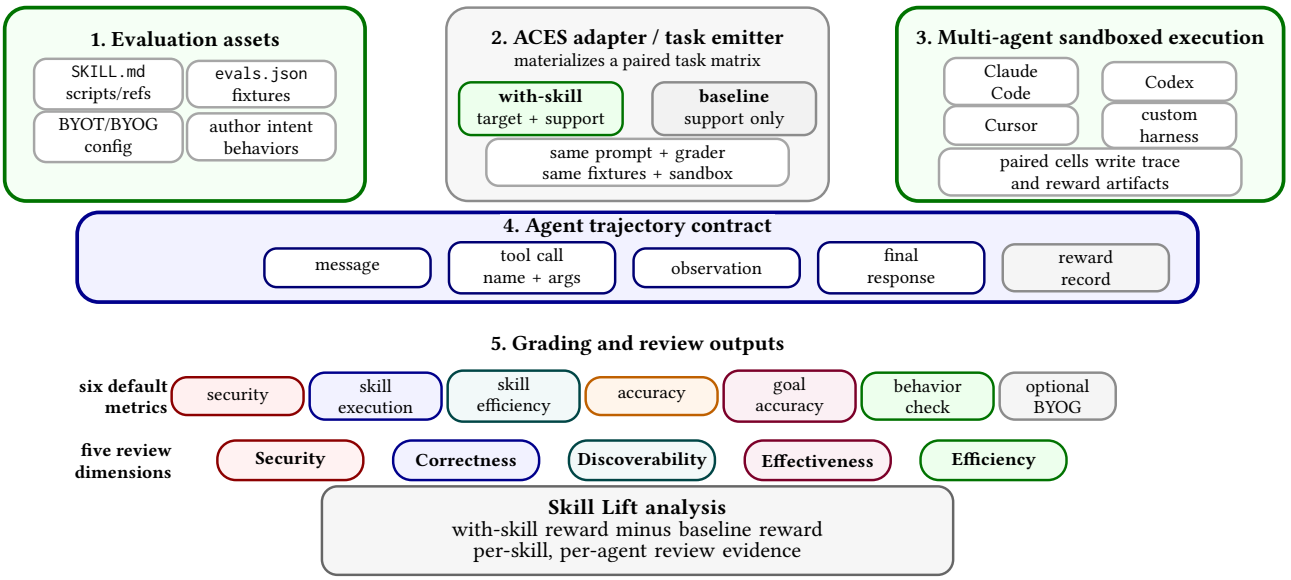

Figure 2. ACES runtime evaluation flow. Evaluation assets feed the ACES adapter and task emitter, which creates paired with-skill and baseline runs while holding configured support skills fixed. Sandboxed agent runs emit or are normalized into ATIF. The same normalized agent trajectory feeds the default metrics, optional BYOG metrics, stakeholder dimensions, and the paired Skill Lift report used for review.

defaults with `EVAL.md`. BYOT and BYOG extend the same contract to product-specific tasks and graders; the paired protocol, trajectory capture, aggregation, and reporting remain unchanged. `expected_behavior` is deliberately a process metric, not a substitute for task success. It is appropriate for observable workflow, safety, or product-specific assertions, but it can over-reward obedience to an author's preferred path if the assertions are too prescriptive. We therefore report behavior check separately from accuracy and goal accuracy, retain the underlying assertion-level evidence, and treat assertion phrasing and judge choice as calibration points for future human-audited studies.

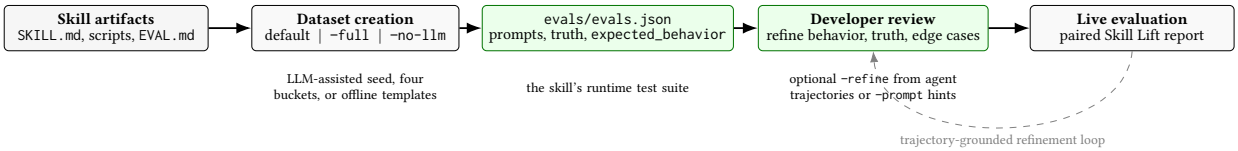

Figure 3. Evaluation-dataset creation and refinement. Skill authors create or update `evals/evals.json` while building the skill: they can write it directly, bootstrap it from skill artifacts, refine it with author intent or saved agent trajectories, and then use it for paired Skill Lift evaluation.

Evaluator suite and Skill Lift. The default evaluator suite has six metrics: security, skill execution, skill efficiency, accuracy, goal accuracy, and behavior check. These map to five stakeholder review dimensions: Security, Correctness, Discoverability, Effectiveness, and Efficiency; BYOG metrics are reported separately when present. Table 1 makes the default metric contract explicit. Every default metric is normalized to $[0, 1]$, where 1 is best. Unless a repository config overrides the policy, the default overall score is the unweighted mean of the available default metrics; each metric therefore has weight $1/6$ when all six are emitted. Stakeholder dimensions are review views over the same normalized metrics, not a second hidden weighting layer.

For task case i, metric m, skill s, and agent harness a, the paired metric delta is

$$\Delta_{s,a,i,m} = S^{\text{with}}_{s,a,i,m} - S^{\text{base}}_{s,a,i,m}.$$

Only task cases with both with-skill and baseline scores for the same metric enter that metric's paired estimate. Missing trajectories, timeouts, or unscored conditions are counted in the evidence inventory and debugging reports, but excluded from paired-case means rather than imputed as failures. A skill-agent cell aggregates first over scored task cases, then over repeated trials:

$$\text{Lift}_{s,a,m} = \frac{1}{|I_{s,a,m}|} \sum_{i \in I_{s,a,m}} \Delta_{s,a,i,m},$$

$$\text{Lift}_{s,a} = \frac{1}{|M_{s,a}|} \sum_{m \in M_{s,a}} \text{Lift}_{s,a,m}.$$

Corpus-level summaries are reported in two views: paired-case means for task-level effect size, and cell/harness summaries for reviewer diagnostics. Positive lift means the skill improves the agent relative to its own matched baseline; negative lift marks a regression target for review.

Trace and execution layer. ACES normalizes harness output into the Agent Trajectory Interchange Format (ATIF): ordered messages, tool calls with arguments, observations, final responses, and reward records. Some harnesses emit structured traces natively; plain-text logs are converted before grading. The ACES adapter translates `evals.json` or BYOT task assets into fresh with-skill and baseline task environments, stages the target plus configured prerequisite/helper/reference/decoy skills, injects a common verifier, and schedules runs across Harbor-compatible backends [13, 14]. Harbor and sandbox runtimes are infrastructure we build on, not the contribution; the ACES contribution is the task-emitter contract, staging policy, ATIF normalization, metrics, aggregation, and review report. Isolation and group workspaces use the same mechanism: isolation measures skill content with only the target skill visible, while group workspaces measure content plus discovery and routing among plausible alternatives.

Baseline construction. ACES does not compare a skilled agent against an empty world. The baseline withholds only the target skill while keeping the agent, model, task prompt, sandbox, scorer, and configured non-target skills fixed. Non-target skills are included by an explicit policy: prerequisite or helper skills are retained when both conditions need them for authentication, shared libraries, or product access; reference decoys come from a fixed small bundle used to test routing pressure; group workspace decoys are sibling or custom skills selected by the workspace experiment. The target skill's instructions, scripts, references, and examples are absent from the baseline. Thus Skill Lift estimates the marginal value of the target skill under a declared workspace configuration, not an intrinsic context-free property of the document. The workspace mode and included support skills are logged with each run so organizations can reproduce or change the baseline policy.

Evaluation-native development. Evaluation assets live with the skill across product repositories, skill-centric repositories, and central registries. Repository automation can run cheap structural checks on every change, optional LLM-judge and security scans when configured, and live paired evaluation when assets or reviewers request it. This mirrors continuous-evaluation practice for AI systems [7, 8, 11, 25, 30] while making skills first-class review artifacts: authors inspect Skill Lift and trajectory evidence, refine datasets from observed behavior, and reviewers approve skill changes with evidence rather than prose alone.

Product-first evaluation. By default, the paired protocol is skill-centered. The same task-emitter contract also supports a product-owned view: a product task suite contains shared prompts, fixtures, expected outcomes, and `expected_`

Table 1. Default ACES metric definitions. All metrics are normalized to $[0, 1]$ before aggregation; higher is better.

Metric	Evidence and scorer	Range and normalization	Default weight and stakeholder view	Known failure modes
Security	Deterministic trace scan over commands, paths, observations, and outputs; optional repository scanners run as separate gates.	1 when no unsafe pattern is found; reduced by detected findings such as secret-like output, destructive commands, or unauthorized paths.	1/6 of default overall; Security.	Pattern false positives/negatives; missing trajectories cannot be fully scanned.
Skill execution	Trajectory evidence for expected skill activation, expected script/tool use, and workflow order.	Fractional completion of execution checks per task case.	1/6; Discoverability and Effectiveness.	Equivalent workflows may be under-credited if not encoded in the dataset.
Skill efficiency	Routing and tool-use evidence: skills read, relevant versus irrelevant skill/tool calls, and productive calls.	Fractional routing/tool-efficiency score per task case.	1/6; Efficiency and Discoverability.	Legitimate exploration can look inefficient; visible-skill count changes routing pressure.
Accuracy	LLM-as-Judge assessment of final answer and trajectory against the task and ground truth.	Judge rubric normalized to $[0, 1]$.	1/6; Correctness.	Sensitive to judge model and ambiguous task wording.
Goal accuracy	Reference-based goal match between the outcome and ground truth.	RAGAS/LLM goal score normalized to $[0, 1]$.	1/6; Correctness and Effectiveness.	Artificially low when a task requires unavailable network or product endpoints.
Behavior check	LLM yes/no judgments of ordered `expected_behavior` assertions against the ATIF trajectory.	Fraction of assertions satisfied.	1/6; Effectiveness, with Security when assertions are safety-specific.	Sensitive to assertion phrasing; can reward author-intended process even when final task success is weak.

`behavior` assertions, while a companion config declares which capability packages to compare. The schema separates the task suite from target materialization:

```
schema_version  1
eval_mode       product
targets[]       {name, type, path?, skills?}
target.type     baseline|skill|skill_bundle
                |team_skills|plugin
harbor          task_source=evals_json
                n_attempts=k
grading         mode=aces_plus_custom.
```

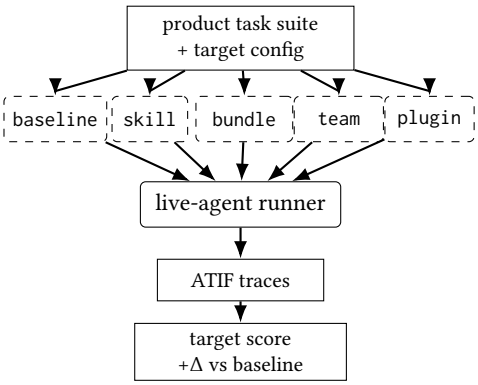

Figure 4. Product-first evaluation reuses the ACES live-agent protocol across product-level targets.

The `baseline` target runs the product task suite without a product skill package. A `skill` target stages one skill; bundle and team targets stage manifest-defined groups; and `plugin` stages plugin-provided skills plus tool or MCP configuration when the product task environment does not already provide it. ACES materializes every selected target into the same live-agent runner, captures ATIF trajectories, and reports validity, target score, optional pass@k, and delta against the product baseline (Figure 4). This view asks whether a capability package helps an agent complete product workflows and whether the member skills cooperate in context. We reserve *Skill Lift* for skill-centered comparisons; bundle, team, and plugin targets are reported as product-target deltas. In this paper, product-first evaluation is an operating mode of the ACES protocol rather than part of the 947-case production-skill headline analysis.

4 Empirical Evaluation

We evaluate ACES on a corpus of 145 real skills drawn from internal enterprise repositories and public skill catalogs. The scan-side results in §2 establish that document checks are useful review gates; this section asks what live trials add. Paired with-skill/baseline runs produce ATIF trajectories, tool-use records, metric deltas, and author-facing findings across harnesses and model variants.

4.1 Corpus

The 145 scanned skills are drawn from internal enterprise skill repositories and public skill catalogs. They span seven category bins: System Access (49 skills), Deployment (41), Platform (29), Data Infra (21), Troubleshooting (3), Dev-Tooling

Table 2. Evidence and filtering inventory for the empirical section.

Item	Count
Skills with Tier 1 and Tier 2 scan results	145
Unique live-eval skill variants exercised	89
Scaling stress variants excluded from headline	25
Production skills in main live inventory	64
Production skills with scored paired cases	58
Primary harnesses in headline	4
Production skill–agent cells in inventory	201
Cells contributing scored paired cases	177
Paired trial files	289
Scored paired task cases	947
Condition runs	578
Reward rows / saved ATIF trajectories	2,077 / 2,022
Tool calls / behavior observations	11,642 / 9,091

(1), and Other (1). Mean structural scores are tightly clustered across categories (78.7–82.1), with an overall mean of 79.2 and median of 79.8. The public-catalog portion includes GPU/ML systems skills for model-training recipes, sequence packing and long-context fine-tuning, parallelism strategy selection, MoE communication overlap and hardware configuration, memory and performance tuning, evaluator launch and configuration, inference serving, code contribution, route optimization, and quadratic-programming formulation.

4.2 Live Subset and Evidence Inventory

Across all live-agent trial artifacts we exercised 89 unique skill variants. For the main production-skill analysis, we separate out the 25 skill/count scaling-study variants and a single non-primary harness row, leaving 64 production skills evaluated on four primary harnesses. Of these, 58 production skills have scored paired task cases. The main subset contains 201 production skill–agent cells in the trial-file inventory; 177 of those cells contribute scored paired task cases. It includes 289 paired trial files, 947 scored paired task cases, 578 condition runs, 2,077 task-level reward rows, and 2,022 saved ATIF trajectories. The 55-row reward/trajectory gap consists of reward rows whose trajectory was missing or could not be reconstructed. The saved trajectories contain 14,335 recorded agent steps, 11,642 tool calls, and 9,091 behavior observations. The paper reports aggregate, anonymized skill identifiers; raw trajectories and logs remain outside the paper repository.

We release aggregate CSVs, anonymized figures, and paper-facing counts. Raw trajectories and logs are retained for verification but not released with the submission because they contain internal hostnames, repository paths, and product identifiers. We therefore treat the paper as an evidence-backed report rather than a public benchmark release.

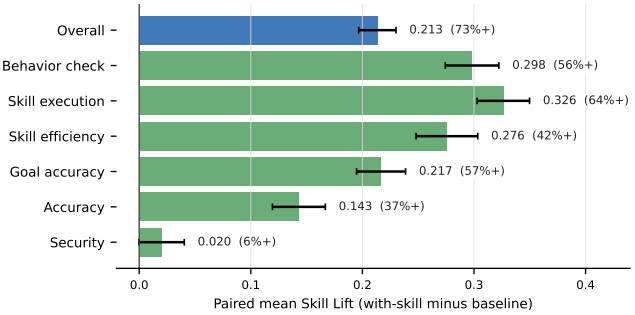

Figure 5. Paired mean Skill Lift for the six default ACES metrics plus overall score, with 95% confidence intervals over 947 paired task cases. Parentheses show the fraction of paired cases with positive lift for that metric.

4.3 Aggregation and Uncertainty

The headline interval in Figure 5 is descriptive: a 95% normal CI over paired task-case deltas. It is useful for reading the plotted effect size but should not be interpreted as 947 independent skills. Cases are clustered by skill, harness, and repeated trial. As a cluster check, bootstrapping over production skills gives an overall lift interval of [0.1898, 0.2350], and bootstrapping over skill–agent cells gives [0.1880, 0.2385]. Both are consistent with the paired-case interval and remain positive. Repeated-run coverage is partial in the 201-cell production trial-file inventory: 88 cells have two paired trial files and the remaining 113 have one. Among the 88 repeated cells, the median within-cell overall-lift standard deviation is 0.0319 (mean 0.0699); single-trial cells are used for effect-size reporting, not for per-cell variance claims.

4.4 Headline Live Result

Across 947 paired task cases from the production-skill subset, the mean paired overall Skill Lift is 0.2134 with a 95% normal CI of [0.1967, 0.2301]. In 72.8% of paired cases, the with-skill condition scores higher than the baseline.

Figure 5 decomposes the lift by the active metric set. The largest mean gains are not only final-answer correctness: skill execution improves by 0.3263, behavior check by 0.2983, and skill efficiency by 0.2758. These trajectory-level signals capture whether the agent read the expected skill, followed the workflow, avoided wasteful routing, and satisfied author-specified behaviors; final-answer accuracy improves too (0.1431), but is not the whole story. Skill efficiency has the third-largest mean lift but positive lift in only 41.7% of paired cases, indicating a high-variance tradeoff in which some runs exchange efficiency for correctness or workflow completion.

4.5 What the Live Report Adds

The live report makes skill value observable in the same way a test report makes code behavior observable: it shows the

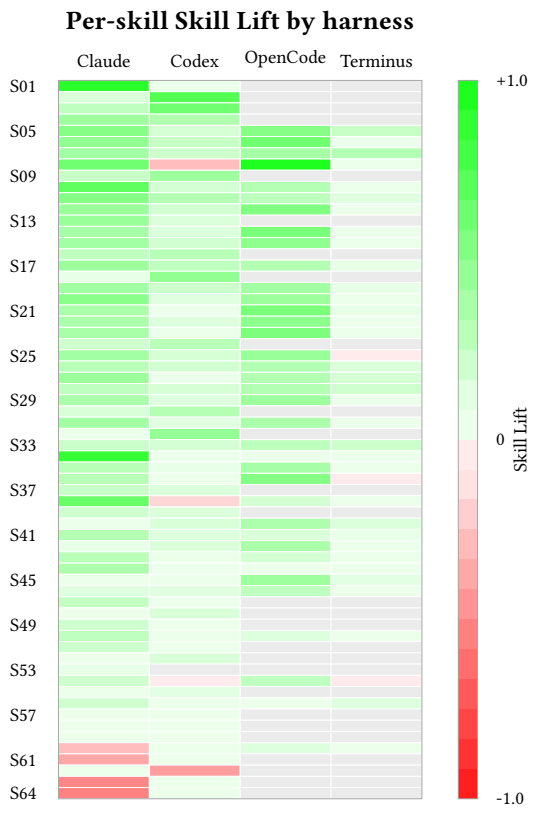

Figure 6. Compact Skill Lift heatmap over the production-skill headline subset and four primary harnesses. Rows are anonymized skill identifiers; color encodes paired overall Skill Lift.

two condition rewards and the delta, per agent. In our completed runs (Figure 7(a)), mean overall lift is positive across the four primary harnesses: 0.3611 for OpenCode, 0.2904 for Claude Code, 0.1264 for Codex, and 0.0896 for Terminus-2. These harness means are diagnostics; the 0.2134 headline is paired-task-case weighted rather than an unweighted mean of harness means. The Codex value, for example, is simply the paired difference on the same Codex task cases: with-skill runs scored 0.6726 on average and matched baseline runs scored 0.5462, so the lift is 0.1264. This is not a claim that Codex is weak; it says how much the skill condition improved over Codex's own baseline in this slice. A document scan cannot produce this view because it never observes routing, execution, or final task outcome.

The compact heatmap in Figure 6 keeps the per-skill view without consuming a full page. Each row is one anonymized skill and each column is one primary harness, making the heterogeneity visible even when aggregate lift is positive.

Figure 7(b) narrows the same paired-lift view to a matched six-skill task subset and labels the default model or backend used by each harness, including OpenCode. These values need not equal panel (a), which aggregates the full production-skill headline subset. Panel (c) separately reruns the Codex harness over the same task subset with different model backends; it should be read as a model-backend sensitivity slice, not as a cross-harness leaderboard. **Model-backend finding (Figure 7(c)).** As shown in Figure 7(c), GPT-5.5 has strong with-skill quality (0.7641), but lower measured lift (0.0506) because its matched baseline is also strong (0.7135). The bars and Δ labels make the interpretation visible: stronger backends can do more without the skill, compressing Δ while preserving high with-skill quality. For public, well-documented workflows, this can happen because the baseline can recover part of the same procedure without the mounted skill. ACES separates absolute model capability from added skill value, exposing model–skill interactions that document scanning cannot.

As a separate stress test, we also ran 25 skill/count variants—five target skills at 1, 5, 10, 20, and 50 visible skills—evaluated as 50 harness cells across two harnesses. This small study suggests stable behavior through 20 visible skills in the sampled setup: mean overall delta stays between 0.133 and 0.149 for 1–20 visible skills, while mean wall time rises from 258s at 1 visible skill to 451s at 20. The with-skill pass rate is 0.725 at 1 visible skill and 0.55 at 50 visible skills, where wall time rises to 1290s, so we treat the high-count setting as a latency and routing stress condition rather than part of the production headline; we do not mix those stress-test variants into the production-skill headline above.

4.6 Negative Lift as a Debugging Signal

A positive aggregate does not mean every skill helps every agent. Across the 947 paired task cases, 87 have negative overall lift; in the 201-cell production trial-file aggregate, 11 cells have negative mean lift and 11 have zero mean lift at reported precision. These are review targets: paired evaluation turns a regression into a traceable comparison between the with-skill and baseline runs.

The trajectories show two distinct classes. Some negative cells reflect execution instability rather than a substantive skill failure: for example, one with-skill run produced no scored trials while the baseline completed. Clearer-cut negative cases show the skill changing the agent's behavior in the wrong direction: the agent found or attempted to read the skill, but produced a truncated or meta-level response, skipped verification, or spent extra tool calls without improving the answer. In one anonymized technical-configuration case, activation and some behavior checks improved, but goal accuracy and efficiency fell enough to make overall lift negative. Live evaluation therefore separates "never discovered" from "discovered but misused," both invisible to document scanning.

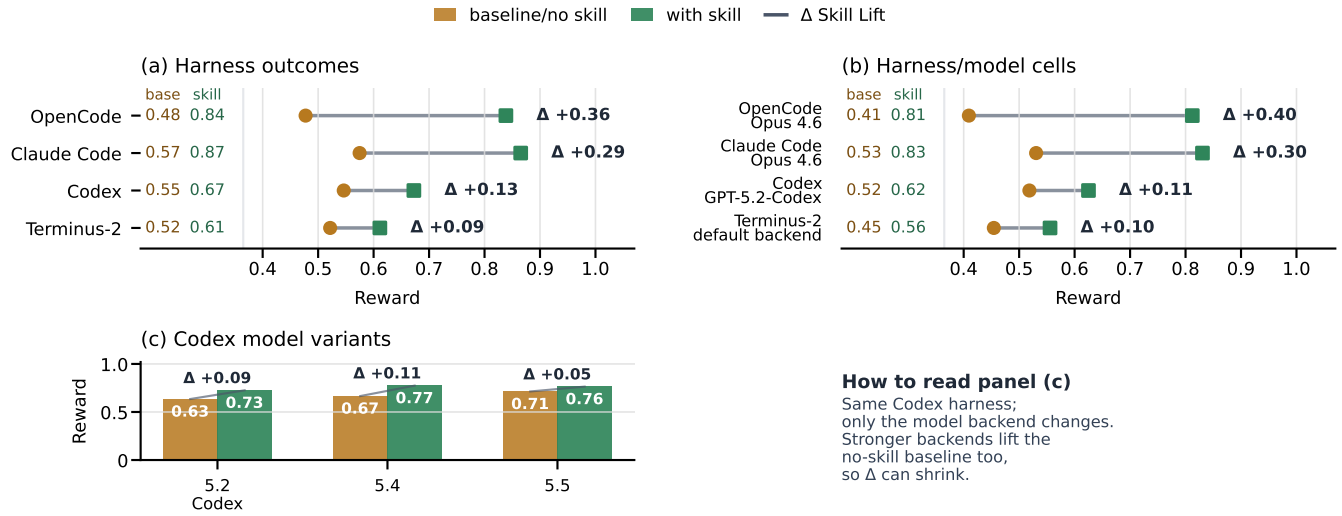

Figure 7. How Skill Lift appears in a live evaluation report. (a) Full harness aggregate. (b) Matched six-skill subset grouped by harness/model cell. (c) Codex-only model-variant slice: stronger backends can raise the baseline/no-skill bar, shrinking Δ even when with-skill reward remains high. Panel (b) labels the configured backend; OpenCode was run with Opus 4.6. Reward labels are rounded to two decimals, while Δ is computed from full precision. Here Δ Skill Lift is with-skill minus baseline/no-skill reward.

4.7 Static Scores Are Not Runtime Evidence

Static scores remain valuable for fast review, but they should not be treated as runtime evidence. Of the 64 production skills in the trial-file inventory, 62 have matching doc-scan metadata. On those 62 skills, Tier 1 structural score has Spearman $\rho = -0.0181$ against overall live lift (Fisher-z approximate 95% CI [-0.2667, 0.2327]), and Tier 2 LLM-judge score has Spearman $\rho = -0.0266$ (95% CI [-0.2745, 0.2247]). These correlations are statistically indistinguishable from zero. The live run instead records 2,022 trajectories, 11,642 tool calls, and 9,091 behavior observations: evidence of discovery, selection, workflow execution, recovery, `expected_behavior` satisfaction, and safety.

4.8 Trace-Grounded Qualitative Readout

Saved trajectories also give reviewers a qualitative readout that aggregate lift cannot provide. In the live artifacts, a public technical workflow skill showed the positive mechanism directly: the agent activated the expected skill, used the intended tool path, made 1/1 productive tool calls, and satisfied 5/5 expected behaviors. Other traces revealed partial success that final answers hid: an authentication workflow diagnosed missing OAuth tokens but skipped the authorization URL and did not rerun verification, and a multi-step artifact workflow exposed missing polling and retry guidance. Security backfill likewise found secret-like strings, destructive-command patterns, unauthorized-path access, and missing-trajectory cases, supporting the decision to keep security separate from behavior check.

5 Discussion and Limitations

Our results come from a 145-skill corpus spanning internal enterprise skills and public skills from community catalogs; for the live-agent evaluation, we use four primary agent harnesses with uneven cell coverage. Replicating on skills authored by additional organizations and on harnesses we do not currently support would strengthen the generalizability claim. The corpus is also skewed toward System Access, Deployment, Platform, and Data Infra skills; we do not claim the same lift distribution for underrepresented categories such as Troubleshooting, Dev-Tooling, or creative skills. The weak Spearman $\rho = 0.14$ between scan methods is evidence of complementary review signals in this corpus, and deserves replication on additional populations. More directly, the near-zero correlations between scan scores and live lift (Tier 1 $\rho = -0.0181$, Tier 2 $\rho = -0.0266$) are consistent with no useful monotonic runtime proxy in these scan scores, rather than evidence of a negative relationship.

Causal interpretation. The paired design holds the task, harness, model, scorer, sandbox, and configured non-target skills fixed, so it is stronger than a single-condition live score. It still does not identify an environment-independent "intrinsic" contribution of a skill. Adding or removing a skill can change routing pressure, context allocation, competition with sibling skills, or interactions with prerequisite skills. We therefore interpret Skill Lift as the target skill's marginal contribution under the declared workspace and baseline policy.

Judge-model sensitivity. LLM-judge scoring varies with the judge model; we rotated three judges and report per-judge spread. Absolute LLM-judge values carry judge attribution and should not be compared across corpora without matching the model. Relative ranking within a corpus is preserved.

Model-update caveat. Skill Lift is model- and harness-specific; after model updates, rerun paired evaluations before treating old lift numbers as current evidence.

Partial-credit note for internal-network skills. Several corpus skills reach enterprise-only endpoints behind a VPN that the live-agent container does not cross. Script calls on those skills receive network errors; `goal_accuracy` absolute values on those skills are therefore a lower bound. *Skill Lift* (with minus without) is still informative because both conditions see the same network state, but endpoint unavailability can compress or distort absolute success metrics.

Operational cost and deployment. The scan layer is cheap enough for every change; live runs are heavier because they require sandbox startup, agent execution, evaluator calls, and trajectory storage. In practice we run static checks by default, use LLM-judge and security checks as configured review gates, and run live paired evaluation for release candidates, high-risk skills, or reviewer-requested changes. Reports include token/cost accounting and can be rerun after model or skill updates.

Other caveats. Cost is tracked in tokens; monetary cost depends on the LLM gateway. We report confidence intervals for the headline paired-case lift, but richer subgroup intervals and formal hypothesis tests for negative-lift classes are follow-up work. The live layer includes a trace-level `security` metric, and `expected_behavior` can add skill-specific safety checks; the external static security scanner remains an integration for supply-chain and prompt-injection analysis, not a contribution of this paper.

6 Related Work

SkillsBench [17] and the in-the-wild skill study [20] establish that skills can change agent performance and that retrieval from large skill corpora is itself difficult. They do not make an arbitrary skill artifact the unit of CI review: scoring its document, running author-owned paired tasks, reporting added value, refining datasets from trajectories, and supporting BYOT/BYOG. Agent and tool-use benchmarks such as Terminal-Bench [22], SWE-bench [15], AgentBench [18], GTA-2 [28], MCP-Bench [29], and MCPEval [21] evaluate agents on fixed task sets; ACES instead asks whether a specific skill improves a fixed agent on that skill's own tasks.

Our metrics use established evaluation components. G-Eval [9, 19] motivates structured LLM-as-Judge form filling, RAGAS [10] motivates reference-based goal accuracy, and process-evaluation work shows why final-answer scoring hides skipped steps and tool misuse [12]. Vendor and community skill guidance informs the dataset schema and behavior checks [1, 3–6, 16, 23, 24, 26, 27]. Harbor [13, 14] provides sandboxed execution and ATIF-native interfaces that ACES builds on. Our contribution is the skill-specific composition: the adapter/task-emitter pattern, paired-run protocol, ATIF normalization across harnesses, six default metrics plus BYOG, Skill Lift aggregation, and repository-native reporting.

7 Conclusion and Future Work

Repository review gates for skills often scan the skill document; ACES adds what scanning cannot see in those workflows: a live agent running the skill, compared against a paired baseline. On 145 real skills we find that structural checks and LLM-judge rubrics measure complementary facets (Spearman $\rho = 0.14$), while neither observes runtime behavior. The live layer provides the missing piece: in 947 paired task cases from 64 production skills, with-skill runs improve overall score by 0.2134 on average (95% CI [0.1967, 0.2301]). The largest mean gains appear in skill execution, behavior check, and skill efficiency, giving reviewers evidence about discovery, workflow following, routing, and tool use rather than only final-answer quality. The same paired traces also expose negative-lift cases, where a skill introduces routing overhead, incomplete execution, or harness-specific regressions that a document scan would miss. Paired with-skill and baseline runs are normalized into ATIF, graded by the six-metric ACES default suite and optional domain-specific metrics, mapped into five stakeholder dimensions, and summarized as Skill Lift. The ACES adapter and task emitter stage fresh task environments across agents, task sources, workspace modes, grading modes, and sandbox backends, including isolated and group workspaces for skill-selection and prerequisite-skill scenarios. BYOT and BYOG let teams keep product-specific tasks and graders while ACES supplies the paired-run protocol, trajectory capture, aggregation, and reporting. Repository automation makes the workflow routine on pull or merge requests, turning evaluation assets into the skill analogue of a test suite.

Future work. Follow-ups include longitudinal Skill Lift under model updates, broader validation on external corpora, automated prerequisite/decoy selection, subgroup intervals, and tests for negative-lift classes. We also plan larger multi-attempt/pass@k studies. In an 8-skill Codex/GPT-5.5 pilot with three stop-on-pass attempts, 24 with-skill first-attempt runs scored below the 0.6 pass threshold. Of those, 17 improved on later attempts, but none crossed the threshold (pass@1=pass@3=25%), showing ACES can measure retry recovery before claiming it improves outcomes.

References

[1] Anthropic. 2025. Agent Skills: Best Practices. Claude Platform Documentation. https://platform.claude.com/docs/en/agents-and-tools/agent-skills/best-practices Accessed April 2026.

[2] Anthropic. 2025. Equipping Agents for the Real World with Agent Skills. Anthropic Engineering Blog. https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills Accessed April 2026.

[3] Anthropic. 2025. How to Create Skills: Key Steps, Limitations, and Examples. Claude Blog. https://claude.com/blog/how-to-create-skills-key-steps-limitations-and-examples Accessed April 2026.

[4] Anthropic. 2025. skill-creator: Building Agent Skills. GitHub repository anthropics/skills. https://github.com/anthropics/skills/tree/main/skills/skill-creator Accessed April 2026.

[5] Anthropic. 2025. Skills Notebooks: Introduction. Claude Platform Cookbook. https://platform.claude.com/cookbook/skills-notebooks-01-skills-introduction Accessed April 2026.

[6] Anthropic. 2025. Teach Claude Your Way of Working Using Skills. Claude Support. https://support.claude.com/en/articles/12580051-teach-claude-your-way-of-working-using-skills Accessed April 2026.

[7] Anthropic. 2026. Demystifying Evals for AI Agents. Anthropic Engineering Blog. https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents Accessed May 2026.

[8] Jialong Chen, Xander Xu, Hu Wei, Chuan Chen, and Bing Zhao. 2026. SWE-CI: Evaluating Agent Capabilities in Maintaining Codebases via Continuous Integration. *arXiv preprint arXiv:2603.03823* (2026). https://arxiv.org/abs/2603.03823

[9] Confident AI. 2024. G-Eval: The Definitive Guide. Confident AI Blog. https://www.confident-ai.com/blog/g-eval-the-definitive-guide Accessed April 2026.

[10] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAS: Automated Evaluation of Retrieval Augmented Generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (EACL): System Demonstrations*. https://arxiv.org/abs/2309.15217

[11] Google Cloud. 2024. MLOps: Continuous Delivery and Automation Pipelines in Machine Learning. Google Cloud Architecture Center. https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning Accessed May 2026.

[12] Milan Gritta, Debjit Paul, Xiaoguang Li, Lifeng Shang, Jun Wang, and Gerasimos Lampouras. 2026. Process Evaluation for Agentic Systems. In *Findings of the Association for Computational Linguistics: EACL 2026*. Association for Computational Linguistics, Rabat, Morocco, 2678–2692. doi:10.18653/v1/2026.findings-eacl.140

[13] Harbor Framework. 2026. Harbor: A Framework for Sandboxed Agent Task Evaluation. Project homepage. https://www.harborframework.com/ Accessed April 2026.

[14] Harbor Framework. 2026. Harbor BaseAgent API Reference. Harbor documentation. https://mintlify.wiki/harbor-framework/harbor/api/base-agent Accessed April 2026.

[15] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?. In *The Twelfth International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2310.06770

[16] LangChain. 2025. Evaluating Skills. LangChain Blog. https://www.langchain.com/blog/evaluating-skills Accessed April 2026.

[17] Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, Shuyi Wang, Binxu Li, Qunhong Zeng, Di Wang, Xuandong Zhao, Yuanli Wang, Roey Ben Chaim, Zonglin Di, Yipeng Gao, Junwei He, Yizhuo He, Liqiang Jing, Luyang Kong, Xin Lan, Jiachen Li, Songlin Li, Yijiang Li, Yueqian Lin, Xinyi Liu, Xuanqing Liu, Haoran Lyu, Ze Ma, Bowei Wang, Runhui Wang, Tianyu Wang, Wengao Ye, Yue Zhang, Hanwen Xing, Yiqi Xue, Steven Dillmann, and Han chung Lee. 2026. SkillsBench: Benchmarking How Well Agent Skills Work Across Diverse Tasks. *arXiv preprint arXiv:2602.12670* (2026). https://arxiv.org/abs/2602.12670

[18] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2024. AgentBench: Evaluating LLMs as Agents. In *The Twelfth International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2308.03688

[19] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2511–2522. https://aclanthology.org/2023.emnlp-main.153

[20] Yujian Liu, Jiabao Ji, Li An, Tommi Jaakkola, Yang Zhang, and Shiyu Chang. 2026. How Well Do Agentic Skills Work in the Wild: Benchmarking LLM Skill Usage in Realistic Settings. *arXiv preprint arXiv:2604.04323* (2026). https://arxiv.org/abs/2604.04323

[21] Zhiwei Liu, Jielin Qiu, Shiyu Wang, Jianguo Zhang, Zuxin Liu, Roshan Ram, Haolin Chen, Weiran Yao, Shelby Heinecke, Silvio Savarese, Huan Wang, and Caiming Xiong. 2025. MCPEval: Automatic MCP-based Deep Evaluation for AI Agent Models. *arXiv preprint arXiv:2507.12806* (2025). https://arxiv.org/abs/2507.12806

[22] Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjorn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. 2026. Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces. *arXiv preprint arXiv:2601.11868* (2026). https://arxiv.org/abs/2601.11868

[23] OpenAI. 2025. Codex Skills. OpenAI Codex Documentation. https://developers.openai.com/codex/skills Accessed April 2026.

[24] OpenAI. 2025. Evaluating Skills. OpenAI Developers Blog. https://developers.openai.com/blog/eval-skills/ Accessed April 2026.

[25] OpenAI. 2026. Evaluation Best Practices. OpenAI API Documentation. https://platform.openai.com/docs/guides/evaluation-best-practices Accessed May 2026.

[26] Spillwave. 2025. Mastering Agentic Skills: The Complete Guide to Building Effective Agent Skills. Spillwave Publication. https://pub.spillwave.com/mastering-agentic-skills-the-complete-guide-to-building-effective-agent-skills-d3fe57a058f1 Accessed April 2026.

[27] Vercel. 2025. Agent Skills Explained: An FAQ. Vercel Blog. https://vercel.com/blog/agent-skills-explained-an-faq Accessed April 2026.

[28] Jize Wang, Xuanxuan Liu, Yining Li, Songyang Zhang, Yijun Wang, Zifei Shan, Xinyi Le, Cailian Chen, Xinping Guan, and Dacheng Tao. 2026. GTA-2: Benchmarking General Tool Agents from Atomic Tool-Use to Open-Ended Workflows. *arXiv preprint arXiv:2604.15715* (2026). https://arxiv.org/abs/2604.15715

[29] Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Shah, Yujia Bao, and Eugene Siow. 2025. MCP-Bench: Benchmarking Tool-Using LLM Agents with Complex Real-World Tasks via MCP Servers. *arXiv preprint arXiv:2508.20453* (2025). https://arxiv.org/abs/2508.20453

[30] Boming Xia, Qinghua Lu, Liming Zhu, Zhenchang Xing, Dehai Zhao, and Hao Zhang. 2024. An Evaluation-Driven Approach to Designing LLM Agents: Process and Architecture. *arXiv preprint arXiv:2411.13768* (2024). https://arxiv.org/abs/2411.13768