COMMERCEAGENTBENCH: BENCHMARKING LONG-HORIZON AGENTS IN HIGH-FIDELITY, STATEFUL, AND REPRODUCIBLE REPLICAS OF REAL ONLINE SERVICES

Anonymous authors
Paper under double-blind review

ABSTRACT

Whether large language model agents can carry out sustained operational work remains a central evaluation challenge. Such work spans tools and systems, changes shared state, and depends on repeated judgment. CommerceAgentBench (CAB) evaluates this ability in high-fidelity, stateful, reproducible replicas of real services across browser UI, API/MCP, isolated command-line, replay, and open-source appliance forms. Task-surface fidelity aligns the declared routes, controls, validations, visible outputs, and state transitions with upstream evidence and regression tests. We evaluate a fixed set of 107 cases spanning commerce, sourcing, office suites, marketplace operations, logistics, and ERP-like reconciliation. Human authors reconstruct workflows from request examples or curated procedures, ground workspaces in public or officially sourced synthetic records, and define outcome contracts. Agent actions mutate durable state, and verifiers read final state or artifacts. Scoring combines strict workflow completion with macro-averaged capacity over 834 checks: 799 non-judge checks and 35 grounded boolean model-judge checks. Thirteen models run on three agent frameworks with three independent runs per model–framework pair. The strongest model achieves a 59.7% mean pass rate and 0.892 capacity across frameworks. On a selected 15-cell diagnostic panel, supplying object-selection decisions or target-system state enables previously failed workflows to complete. A 14-verifier state audit and one-change replays on 12 of 32 HARD cases test score responses to controlled errors. These results characterize complete workflows, partial progress, and the effect of targeted interventions. Our data and code are available at `https://anonymous.4open.science/r/CommerceAgentBench-5E42`.

1 INTRODUCTION

Recent agents can write code, search the web, and call tools with growing reliability, but the harder question is whether they can finish a long piece of real work (Jimenez et al., 2024; Huang et al., 2023; Chan et al., 2024; Patil et al., 2023; Qin et al., 2024; Mialon et al., 2024). Operational work exposes this problem at useful scale. A single request may require an agent to read messy state, decide which objects qualify, update records across storefronts, email, documents, SaaS trackers, command-line tools, and ERP-like data sources, and preserve constraints while later steps depend on earlier writes (Yao et al., 2022; Zhou et al., 2023; Drouin et al., 2024; Trivedi et al., 2024; Xu et al., 2025).

Realistic evaluation needs three properties at once. First, the agent must operate through the surfaces used in the workflow. Second, writes must change what later reads and final verifiers observe: an archive, refund, field edit, or status transition becomes part of the state on which subsequent decisions depend. Third, the world must be resettable, inspectable, and replayable so models can be compared from the same initial condition (Zhou et al., 2023; Xie et al., 2024; Trivedi et al., 2024; Lu et al., 2025; Yao et al., 2024).

CommerceAgentBench combines these properties. Each case runs in a replica grounded in captured browser pages, offline assets, upstream documentation or source, replay snapshots, real workflow evidence, or an open-source appliance. Browser UI, API/MCP, daemon-isolated CLI, replay, and OSS surfaces use alignment tests appropriate to their interface. *Task-surface fidelity* aligns the declared

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

routes, controls, validations, visible results, and state transitions with upstream evidence. The agent operates on realistic evidence, commits changes, and encounters their consequences throughout a task.

Task construction is human-led. Authors reconstruct the business intent and completion criteria from request examples or curated operational workflows, assemble an evidence-backed workspace, and write the hidden outcome contract and verifier. LLMs assist drafting and normalization; human authors own the source mapping and admission decision. Each case therefore joins a user request, an operable world, and a verifiable outcome.

This paper makes four contributions.

- A fixed 107-case evaluation set for long-horizon operational agents spanning commerce, public web research, office and email suites, SaaS command-line tools, supplier due diligence, logistics, and document workflows.
- Evidence-aligned task-surface replicas across browser UI, API/MCP, CLI, replay, and OSS forms, with validated writes, durable state, reset, audit, and verifier-only inspection.
- A human-led construction protocol connecting workflow reconstruction, evidence-backed workspaces, outcome contracts, controls, and admission review.
- An evaluation of thirteen models across three frameworks, with repeated runs, paired judgment/state ablations, and verifier audits. Binary completion and check-level capacity expose distinct aspects of operational ability.

2 Related Work

2.1 Agent Benchmarks

Agent evaluation spans software engineering (Jimenez et al., 2024; Yang et al., 2024; Xia et al., 2025), machine-learning experimentation (Huang et al., 2023; Chan et al., 2024), web interaction (Lù et al., 2024; Zheng et al., 2024; He et al., 2024), and operating-system control (Xie et al., 2024; Rawles et al., 2025; Kapoor et al., 2024). Tool and API benchmarks examine tool selection and execution (Patil et al., 2023; Qin et al., 2024; Zhuang et al., 2023; Huang et al., 2024; Song et al., 2023); GAIA combines reasoning with information gathering (Mialon et al., 2024), and AgentBench evaluates agents across diverse environments (Liu et al., 2024). CAB focuses on cross-system operational requests whose completion depends on both decisions and durable changes, reporting strict workflow success alongside partial completion.

2.2 Simulated Environments

TextWorld and ScienceWorld provide controlled interactive worlds (Côté et al., 2018; Wang et al., 2022); Sotopia and AgentSims study social interaction and agent behavior (Zhou et al., 2024; Lin et al., 2023). ToolEmu and ToolAlpaca use language models for tool-environment simulation or tool-use data construction (Ruan et al., 2024; Tang et al., 2023). AppWorld, ToolSandbox, and τ-bench support stateful application and tool interactions (Trivedi et al., 2024; Lu et al., 2025; Yao et al., 2024). CAB connects declared service-surface alignment to shared durable state, resettable fixtures, and protected verifier readouts. Its browser, API/MCP, CLI, replay, and open-source forms carry interface-specific alignment evidence (Section 3.2).

2.3 Operational, Commerce, and Web Agents

WebArena, WorkArena, and TheAgentCompany evaluate web and workplace workflows (Zhou et al., 2023; Drouin et al., 2024; Xu et al., 2025). WebShop studies shopping assistance (Yao et al., 2022), while E-Commerce Bench and MerchantBench study persistent seller-side operations and decisions with delayed consequences (Fan et al., 2026; Shi et al., 2026). ST-WebAgentBench evaluates enterprise web tasks under policy constraints (Levy et al., 2026), and X-WebAgentBench examines multilingual planning and interaction (Wang et al., 2025). CAB's evaluation unit is a bounded operational request, such as billing repair or supplier-evidence coordination. Its horizon

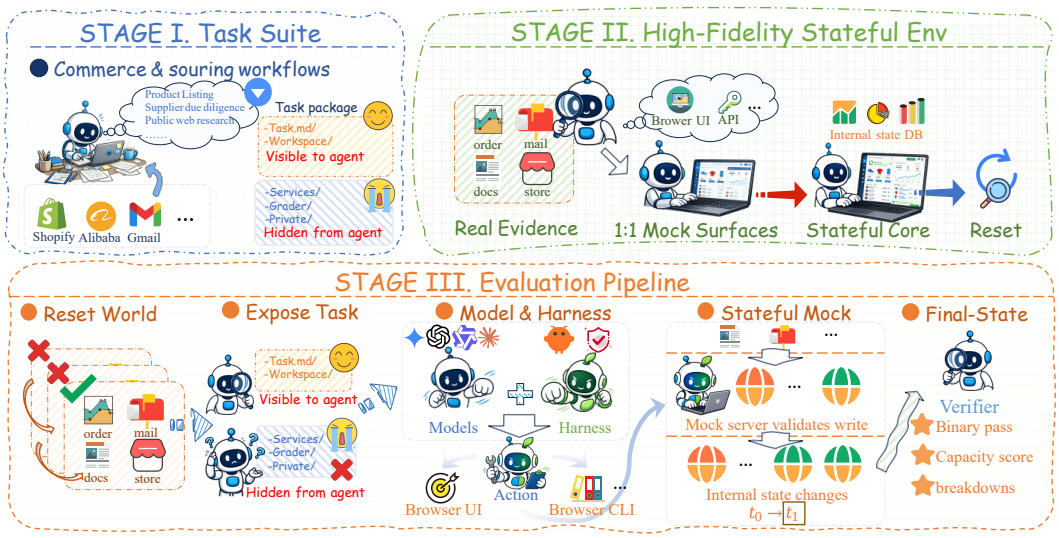

Figure 1: Overview of CommerceAgentBench. A fresh container and allow-listed task copy separate agent-visible inputs from graders and hidden truth. Evidence-backed requests run in stateful replicas across browser UI, API, replay, appliance, and command-line forms. Alignment tests cover declared interactions; validated actions change durable state, and final-state or artifact verifiers score the result.

comes from dependent decisions and writes within a workflow, with contracts specifying required outcomes, object identities, and excluded side effects.

3 The CommerceAgentBench Construction

Figure 1 summarizes CAB: an evidence-backed request and workspace are placed in a fresh environment, the agent acts through aligned service surfaces, and verifiers score the resulting state and artifacts. The construction pipeline connects workflow interpretation, replica implementation, outcome-contract authoring, admission, and replay.

3.1 Case Construction and Admission

CAB uses two authoring routes. In the request-derived route, a curator starts from a traceable request example and records the business intent, target systems, decision points, constraints, and completion criteria. In the workflow-derived route, the starting point is a documented operational procedure and real-world source material. Human authors define the outcome contract and review the finished case; language models assist normalization, translation, and drafting.

Workflow reconstruction turns the request into observable requirements. An online-store diagnosis becomes a read-only contract with real-URL requirements, access-block evidence, screenshots, and a traceability manifest. A four-channel market report becomes a fixed research packet with a normalization policy, recommendation and exclusion decisions, and linked spreadsheet, CSV, JSON, and evidence outputs. These additions preserve the business intent while defining what the agent must deliver.

Evidence-backed workspaces. *Frozen-record cases* use public or authorized CSV pages, reports, announcements, product taxonomies, and API responses with their provenance. *Source-fact-grounded cases* use synthetic records whose schemas, issue types, enums, transitions, and operational facts follow cited official sources. Generated records carry source-fact identifiers that connect each record to its supporting evidence. Privacy transformations preserve cross-record relations, temporal order, and decision boundaries.

Authors remove or consistently replace direct identifiers, account information, local paths, and confidential business values through a human-reviewed mapping. The mapping preserves object

bindings, contradictions, and threshold relationships needed to make the task's decisions. Public execution artifacts remain separate from the raw requests, privacy mappings, and admission records used during authoring.

Each outcome contract specifies accepted final states, required interaction evidence, and excluded side effects. Admission controls exercise valid, incomplete, wrong-binding, collateral-damage, surface-only, and one-change near-correct states against that contract. Leakage scans and visible-only execution review check the agent-facing package. Appendix A gives the source routes, privacy process, and admission procedure; Section 4.4 reports the controlled verifier audit.

3.2 Service Replicas and Stateful Execution

Browser replicas preserve captured pages, DOM structure, localized assets, controls, and task-relevant routes. API and MCP replicas follow upstream documentation, schemas, source, or recorded behavior. Isolated CLI replicas preserve the task's command names, flags, outputs, errors, and state transitions behind a daemon boundary. Replay replicas serve fixed snapshots, while OSS appliances run the corresponding open-source application.

Alignment follows an upstream source-of-truth $\rightarrow$ declared surface $\rightarrow$ implementation $\rightarrow$ regression chain. Snapshot and golden-output tests check appearance and outputs; interaction and negative-path tests check accepted and rejected actions; state-propagation and reset tests check their effects. The 1:1 task-surface designation refers to this declared set of pages, routes, commands, fields, and transitions.

For browser families, captured DOM, open shadow roots, fonts, images, and styles form the page layer; an interaction adapter reconnects controls to a stateful backend. Server-side checks validate required fields and closed-set values, while session tokens identify the permitted write path. A successful create, edit, merge, archive, or publish becomes visible to subsequent page and API reads.

For isolated CLI families, the native command is a thin client that forwards arguments, input, and working directory to a protected daemon. The daemon holds the implementation and per-task state, commits validated mutations, and exposes authenticated state and audit readouts to the harness. Fact sheets record the declared commands, flags, outputs, errors, and deviations, linking the implemented behavior to upstream evidence.

Browser regressions passed 241/241 checks for Shopify v3 core, 21/21 for its storefront, 39/39 for Google Contacts, and 50/50 for Google Docs fidelity, with MCP and Drive regressions of 41/41 and 104/104. CLI regressions passed 540/540 checks across seven families. Six CLI families use versioned fact sheets and golden comparisons; Google Workspace uses documented tool surfaces and regression smoke. Appendix A.3 lists the evidence types, implementations, and documented deviations by family.

When an agent edits a price, archives a message, issues a refund, or updates a document, the server validates the action and commits it to durable state. Later pages, API reads, CLI output, and verifiers observe the same mutation. Each run starts from a fixed fixture, seeded state, or appliance image and archives its trajectory, outputs, logs, and verifier evidence. Author-side snapshot collection and curator-operated seed conversion support replay and new case construction (Appendix A.4).

3.3 Task Suite and Execution Boundary

The reported evaluation set contains 107 cases, selected from a larger task suite by ILP trimming to balance domain and modality coverage and remove near-duplicates. It spans commerce, sourcing, office suites, SaaS tools, marketplace operations, email, documents, logistics, and ERP-like reconciliation. Twenty-eight cases use a browser interface: 24 local replicas and four external web-research cases. The modality split is 67 text-only, 20 vision-required, and 20 browser-text-capable cases; 38 cases use at least one daemon-isolated CLI family. Request provenance determines the request-derived versus workflow-derived label.

Each case packages an agent-visible `task.md` and optional `workspace/`, together with private seeds, truth, and a verifier; 103 cases have an explicit workspace. The harness copies allow-listed public inputs into a fresh container and keeps graders, private state, protected mock implementations, and verifier tokens outside the agent boundary. Static scans, protected paths, authenticated readouts, and execution audits enforce this separation (Appendix A.5).

3.4 Outcome Scoring

For case i, let m_i be its number of required checks and g_{ij} the boolean result of check j. Binary pass requires a completed grader and every required check; capacity measures partial completion and is macro-averaged across cases:

$$\mathrm{Pass}_i = \mathbb{1}[\text{grader completes } \wedge\ m_i > 0\ \wedge\ \forall j,\ g_{ij} = 1]\,, \qquad \mathrm{Cap}_i = \frac{1}{m_i}\sum_{j=1}^{m_i} g_{ij},$$

$$\mathrm{Pass} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{Pass}_i, \qquad \mathrm{Capacity} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{Cap}_i.$$

The evaluation set's rubrics contain 834 checks: 799 non-judge checks (95.8%) and 35 grounded boolean model-judge checks (4.2%). At task level, 101 cases are deterministic-only and six contain judge checks. Deterministic checks read final state, files, or structured artifacts; judge checks evaluate explicit yes/no propositions against expected evidence using `gemini-3.1-pro-preview`.

Rubric authors assign one unit to each distinct decision or outcome capability, collapse repeated formatting checks, and separate independently required batch outcomes. Each task contributes weight $1/N$ to capacity, irrespective of its check count. The granularity migration audit records unchanged binary verdicts; capacity describes the completion fraction under the declared rubric. Appendix A.6 describes verifier revision and harness calibration.

Verifier revision distinguishes an agent capability gap from an ambiguous task contract or a misaligned check. Changes require evidence from the declared contract and controlled states. Positive controls establish that a valid outcome is accepted; negative and one-change controls identify the requirement responsible for rejection, connecting a score to an inspectable state or artifact.

4 Experiments

We evaluate thirteen models on Accio, Pi, and OpenClaw, with three independent runs per model–framework pair: 117 runs of the evaluation set and 12,519 task evaluations. Every run covers the same 107 cases, environments, and final-state verifiers. Vision routing is set explicitly and checked after launch. Table 1 reports means and sample standard deviations. The cross-harness average summarizes each model under these three agent loops; Appendix C gives individual runs and Appendix F gives the reported configuration.

4.1 Main Results

Claude Opus 5 leads the three-harness average with a 59.7% pass rate and 0.892 capacity. On OpenClaw it completes 60.3 of 107 tasks on average (56.4%) and reaches 0.898 capacity. Capacity gives each workflow equal weight and credits its completed requirements; binary pass credits a workflow only when all requirements pass. Their gap records substantial partial completion alongside unfinished workflows.

The average pass rate ranges from 59.7% to 29.2%, with Claude Opus 5 and Claude Opus 4.8 forming the leading pair. Binary pass and capacity distinguish different strengths in the middle of the ranking. DeepSeek V4 Pro ranks sixth by pass rate (47.9%) but tenth by capacity (0.823); DeepSeek V4 Flash completes fewer tasks (44.4%) yet reaches a higher capacity (0.844). One model finishes more workflows, while the other completes more of the requirements across the full suite.

4.2 Repeated Runs and Harness Effects

The OpenClaw columns of Table 1 summarize three complete runs per model (also collected in Appendix B.1). Each run resets the environment to the same initial state, while model sampling and the agent loop vary. Pass-count standard deviations range from 1.5 to 1.7 tasks; capacity standard deviations range from 0.006 to 0.028. The two highest observed mean pass counts differ by 5.6 tasks. GPT-5.5 and Claude Opus 4.7 are much closer, at 49.0 and 48.7, while their capacities are 0.832

Table 1: **Main results** over the 107-case evaluation base. The same tasks, mocks, and final-state verifiers run on three agent frameworks: Accio, Pi, and OpenClaw. Each cell is the mean ± sample standard deviation over three independent runs. **Pass** is the binary count out of 107 and **Cap** is the macro-average of within-task check completion; **Average** is the mean pass rate (%) and mean capacity across the three harnesses. Best per column in bold.

(a) Binary pass count (out of 107) and average pass rate

Model	Accio	Pi	OpenClaw	Average (%)
Claude Opus 5	$\mathbf{66.3}_{\pm 1.5}$	$\mathbf{65.0}_{\pm 1.0}$	$\mathbf{60.3}_{\pm 1.5}$	**59.7**
Claude Opus 4.8	$59.3_{\pm 1.5}$	$56.3_{\pm 1.5}$	$54.7_{\pm 1.5}$	53.1
GPT-5.6 Sol	$55.3_{\pm 1.5}$	$54.0_{\pm 1.7}$	$51.3_{\pm 1.5}$	50.1
Claude Opus 4.7	$56.3_{\pm 1.5}$	$52.3_{\pm 1.5}$	$48.7_{\pm 1.5}$	49.0
Qwen 3.8 Max	$56.0_{\pm 2.0}$	$53.3_{\pm 1.5}$	$47.3_{\pm 1.5}$	48.8
DeepSeek V4 Pro	$54.3_{\pm 1.5}$	$52.7_{\pm 1.5}$	$46.7_{\pm 1.5}$	47.9
GPT-5.5	$50.0_{\pm 1.7}$	$49.0_{\pm 1.7}$	$49.0_{\pm 1.7}$	46.1
Gemini 3.6 Flash	$51.0_{\pm 1.0}$	$49.0_{\pm 1.7}$	$46.3_{\pm 1.5}$	45.6
GLM 5.2	$51.0_{\pm 1.0}$	$50.0_{\pm 1.0}$	$42.3_{\pm 1.5}$	44.7
DeepSeek V4 Flash	$50.3_{\pm 1.5}$	$47.7_{\pm 1.5}$	$44.7_{\pm 1.5}$	44.4
Gemini 3.5 Flash	$46.3_{\pm 1.5}$	$43.7_{\pm 1.5}$	$39.3_{\pm 1.5}$	40.3
GPT-5.6 Luna	$48.0_{\pm 2.0}$	$42.3_{\pm 1.5}$	$36.3_{\pm 1.5}$	39.5
Gemini 3 Flash	$32.7_{\pm 1.5}$	$31.7_{\pm 0.6}$	$29.3_{\pm 1.5}$	29.2

(b) Capacity score

Model	Accio	Pi	OpenClaw	Average
Claude Opus 5	$0.876_{\pm 0.018}$	$\mathbf{0.901}_{\pm 0.004}$	$\mathbf{0.898}_{\pm 0.006}$	**0.892**
Claude Opus 4.8	$\mathbf{0.886}_{\pm 0.012}$	$0.900_{\pm 0.006}$	$0.854_{\pm 0.006}$	0.880
GPT-5.6 Sol	$0.884_{\pm 0.012}$	$0.892_{\pm 0.016}$	$0.855_{\pm 0.012}$	0.877
Claude Opus 4.7	$0.878_{\pm 0.012}$	$0.866_{\pm 0.011}$	$0.871_{\pm 0.012}$	0.872
Qwen 3.8 Max	$0.878_{\pm 0.008}$	$0.856_{\pm 0.011}$	$0.810_{\pm 0.006}$	0.848
DeepSeek V4 Pro	$0.822_{\pm 0.030}$	$0.898_{\pm 0.018}$	$0.749_{\pm 0.012}$	0.823
GPT-5.5	$0.858_{\pm 0.022}$	$0.861_{\pm 0.033}$	$0.832_{\pm 0.016}$	0.850
Gemini 3.6 Flash	$0.813_{\pm 0.016}$	$0.870_{\pm 0.021}$	$0.838_{\pm 0.028}$	0.840
GLM 5.2	$0.796_{\pm 0.010}$	$0.798_{\pm 0.005}$	$0.821_{\pm 0.006}$	0.805
DeepSeek V4 Flash	$0.857_{\pm 0.023}$	$0.848_{\pm 0.033}$	$0.827_{\pm 0.012}$	0.844
Gemini 3.5 Flash	$0.824_{\pm 0.008}$	$0.847_{\pm 0.012}$	$0.802_{\pm 0.007}$	0.824
GPT-5.6 Luna	$0.805_{\pm 0.007}$	$0.812_{\pm 0.006}$	$0.783_{\pm 0.015}$	0.800
Gemini 3 Flash	$0.762_{\pm 0.006}$	$0.787_{\pm 0.027}$	$0.744_{\pm 0.012}$	0.764

and 0.871. The table reports run variation, and the two metrics expose different completion patterns among nearby models.

Harness choice changes the middle of the ranking. Claude Opus 5 and Claude Opus 4.8 hold the first two binary positions on all three frameworks, and Gemini 3 Flash remains last. Claude Opus 4.7 ranks third on Accio, whereas GPT-5.6 Sol ranks third on Pi and OpenClaw. Mean pass counts for a model differ by as much as 11.7 tasks across frameworks. Accio generally produces the highest counts and OpenClaw the lowest, even though the task, environment, and verifier remain fixed.

Capacity also varies with the agent loop. Claude Opus 4.8 leads on Accio, while Claude Opus 5 leads on Pi and OpenClaw. DeepSeek V4 Pro reaches 0.898 on Pi and 0.749 on OpenClaw. These are model–harness results: the task packages and graders are shared, while the harness controls how a model plans, manages context, and executes tools. The observed leading binary order is stable across the three harnesses, while nearby models change order.

4.3 Ablation: Judgment and State Propagation

The intervention study examines 15 selected diagnostic cells: four A1 cells supply the object-selection decision and eleven A3 cells supply target-system state. Each variant keeps the source task, model, tool surface, budget, final goal, and verifier fixed. The agent performs the remaining execution and satisfies side-effect and exclusion constraints. Claude Opus 4.8, GPT-5.5, and Gemini 3 Flash run each variant three times. A flip records at least one successful run in those three attempts; all source tasks have zero recorded passes in this panel. Tables 2–4 summarize the interventions and flips; Appendix B.3 provides check-level results.

Table 2: A1/A3 interventions on 15 selected cells, with three models and three runs per variant. Source tasks have zero recorded passes; a flip is at least one successful intervention run in three attempts. A1 supplies object selection, A3 supplies target-system state, and A2 supplies cleaned evidence. Flip counts fail-to-pass cell–model pairs; Cap. is the simplified-variant capacity.

Ablation	Hands over	Source	Flip	Cap.
A1 judgment	which objects/classes	0/12	**8/12**	0.92
A3 state	target-system state	0/33	**22/33**	0.93
A2 parsing	cleaned evidence	—	≈0	—

Supplying the A1 decision yields at least one completed run in 8 of 12 cell–model pairs; supplying A3 target-system state does so in 22 of 33. The A2 cleaned-evidence controls produce no binary flips. The reported statistic measures completion within three attempts on the selected panel. The retained constraints require the agent to turn the supplied decision or state into an acceptable final outcome.

Table 3: Ablation outcomes by model. Each model runs four A1 and eleven A3 cells. Counts are best-of-three fail-to-pass flips; Cap. is capacity on the simplified variants.

Model	A1 (/4)	A3 (/11)	Total (/15)	Cap.
Claude Opus 4.8	4	8	**12**	0.98
GPT-5.5	3	8	**11**	0.97
Gemini 3 Flash	1	6	**7**	0.89

Table 3 breaks the outcomes down by model. Claude Opus 4.8 flips 12 of 15 cells, GPT-5.5 flips 11, and Gemini 3 Flash flips seven. Their reported simplified-variant capacities are 0.98, 0.97, and 0.89. With object selection supplied, Claude Opus 4.8 completes all four A1 cells and Gemini 3 Flash completes one. GPT-5.5 and Claude Opus 4.8 each complete eight A3 cells after the state intervention.

Table 4 locates these outcomes across API, file, and command-line workflows. A cell is *robust* when at least two models flip, *weak* when one flips, and *no flip* when none does; these labels count agreement across the tested models. Eleven cells are robust, two are weak, and two have no binary flip. Jira triage, Notion evidence upload, Stripe billing repair, and Todoist scheduling each complete for all three models in at least one A3 attempt.

Two A3 cases retain an unmet requirement after intervention: export-control relabeling reaches 7/8 checks and trade-finance processing reaches 8/9. A3 supplies target-system state while leaving the source-system and unrelated-object constraints to the agent. Appendix B.3 reports every cell's check count, best completion fraction, and family-level outcomes (Tables 14 and 15).

4.4 Scoring Validity

The verifier audit measures score responses to controlled state changes in 14 selected verifiers. Its controls exercise correct, partial, surface-only, wrong-identity, collateral, equivalent-final-state, and tolerance-boundary conditions. One-change perturbations on 12 of the 32 HARD cases provide a separate test of near-correct outputs.

Table 4: Per-cell ablation matrix: ✓ denotes a fail-to-pass flip and ✗ no flip, using best-of-three outcomes. A cell is robust when at least two models flip, weak when one flips, and no flip when none does. The panel contains 11 robust, 2 weak, and 2 no-flip cells.

Cell (type)	O4.8	G5.5	Gem
asia-supplier-tour (A1)	✓	✓	✗
google-trends-quality (A1)	✓	✓	✓
cargo-insurance-alloc (A1)	✓	✗	✗
lc-eco-tote-discrepancy (A1)	✓	✓	✗
jira-defect-triage (A3)	✓	✓	✓
notion-evidence-upload (A3)	✓	✓	✓
stripe-billing-repair (A3)	✓	✓	✓
todoist-sourcing (A3)	✓	✓	✓
xfer-dual-source-recon (A3)	✗	✗	✓
xfer-export-control (A3)	✗	✗	✗
xfer-medtech-dossier (A3)	✓	✓	✗
xfer-tiered-approval (A3)	✓	✓	✓
xfer-trade-finance (A3)	✗	✗	✗
xfer-zibo-evidence (A3)	✓	✓	✗
xfer-invoice-followup (A3)	✓	✓	✗
Total / 15	**12**	**11**	**7**

Table 5: Final-state audit of 14 selected verifiers across seven conditions. Observed counts retain all 14 verifiers; the last column gives readout conditions. Equivalent-state acceptance follows the final-state and event requirements of each contract.

Synthetic state	Criterion	Observed	Coverage note
Correct state	pass	14/14 pass	Full target state accepted.
Partial write	fail	14/14 fail	Incomplete final states rejected.
Surface only, no write	fail	14/14 fail	Visible artifacts alone do not earn credit.
Wrong identity binding	fail	14/14 fail	Correct values on wrong objects rejected.
Collateral damage	fail	11/14 fail	Observable for stateful mocks; scope-limited for pure-file and forward-only readouts.
Equivalent final state	contract	9/14 pass	Pure-state readouts accept; event and audit readouts also require the contracted evidence.
Tolerance boundary	both	calibrated	At 0.5 tolerance, $+0.50$ passes and $+0.51$ fails.

All 14 verifiers accept the correct state and reject partial writes, surface-only completion, and wrong-identity bindings (Table 5). Collateral perturbations are rejected by 11 of 14 verifiers; the remaining readouts have narrower state visibility. Equivalent final states pass in nine verifiers. The other five require event or audit evidence in addition to the final values. At a tolerance of 0.5, a deviation of 0.50 passes and 0.51 fails.

Table 6: The 14 audited verifiers grouped by readout layer. Pure-state layers accept equivalent legal final states; event and audit layers also check the required interaction evidence. All four layers reject wrong-identity bindings.

Verifier layer	#	Equiv. path	Side-effect	Bind.
pure-file readout	2	accept	scope-limited	reject
live mock state	7	accept	detect	reject
event / access-log	4	needs event	partial	reject
audit-trail gated	1	method-valid	—	reject

Table 6 relates these outcomes to the verifier interface. Two verifiers read pure files, seven read live mock state, four use event or access logs, and one uses a gated audit trail. The nine pure-state verifiers accept equivalent outcomes reached through different legal paths. Event- and audit-based tasks additionally require the specified interaction evidence. These layers let a task contract score both the committed result and, when required, the operation that produced it.

Table 7: One-change replay on 12 of 32 HARD cases. Every replay fails exactly the attacked check and passes the remaining $N-1$ of N checks. Rows show the perturbation types and the affected check.

Perturbation	**Example**	**Fails (and only)**
extra-token alias	`Highest`→`Highest Priority`	`priority_correct`
confusable id	`RMA-6207`→`RMA-62O7`	`summary_resolved`
substring brand	`navigator`→`lift-away`	`sub_floor_correct`
near-neighbor field	`on_board`→`ship_window`	`material_found`
id-prefix neighbor	`LINNEA`→`LINNEA-EXPORT`	`customer_settled`
wrong source value	discharge → loading amount	`field_resolution`
plural/suffix alias	`material-gap`→`materials-gap`	label/file check

Each one-change replay modifies a confusable alias, identifier, field, or binding and preserves the rest of the state (Table 7). All 12 replays fail exactly the attacked check and retain the other $N-1$ checks. The scores separate the injected semantic error from the correctly completed parts of each audited workflow.

4.5 Workflow Completion Patterns

The OpenClaw case studies illustrate incomplete workflows with substantial partial progress. `cli-stripe-billing-audit-repair` has no binary pass across models and mean capacity 0.87 over 13 checks; each of the three ablation models reaches 13/13 in at least one A3 run. `cli-todoist-sourcing-schedule` has mean capacity 0.83 across 75 checks, and each ablation model reaches 75/75 after A3. Both tasks combine multiple required writes, and their observed completion changes when target-system state is supplied.

On `api-gmail-supplier-rfq-triage`, one OpenClaw run gives Claude Opus 4.7 a full 23/23, Gemini 3 Flash 22/23, and GPT-5.5 20/23; Claude Opus 4.8 and Gemini 3.5 Flash each score 0/23. Capacity distinguishes nearly finished work from runs with no completed check, while binary pass records whether the agent completed the entire request.

The freight-booking variants carry 28–31 checks and reach full capacity for every model tested on them. They require a multi-step interface flow with few open-ended decisions. Their completion contrasts with the Stripe and Todoist cases: check count alone does not determine the observed difficulty. The case evidence distinguishes final-state completion, partially satisfied contracts, and outcomes under targeted intervention. Appendix E gives the detailed cases; Appendix B.2 records an external model-release comparison.

5 Conclusion

CommerceAgentBench evaluates sustained operational work in high-fidelity, stateful, reproducible replicas of real services. Human-authored requests and outcome contracts connect evidence-backed workspaces to aligned browser, API, and command-line surfaces. Validated actions change durable state, and final-state and artifact verifiers measure both complete workflows and partial capability.

Across thirteen models and three frameworks on a fixed 107-case evaluation set, the strongest model reaches a 59.7% mean workflow pass rate and 0.892 macro-averaged capacity. Diagnostic interventions produce completed workflows after supplying object-selection decisions or target-system state, while controlled audits measure verifier responses to semantic errors. These results connect complete and partial outcomes to specific models, agent loops, and interventions.

AI USE STATEMENT

Language models assist task construction through normalization, translation, and drafting. Human authors review the workflow interpretation, source mapping, outcome contract, and admission decision. Evaluation uses `gemini-3.1-pro-preview` for 35 grounded boolean checks across six cases, as specified in Section 3.4. Generative AI also provided feedback on evaluation methodology and assisted interpretation of the reported results. The authors take responsibility for the final manuscript, benchmark artifacts, and AI-assisted content.

ETHICS STATEMENT

This work evaluates software agents in reconstructed service environments. All human annotation and review were performed by the authors; no external annotators were recruited. Operational writes take place in the benchmark's task environments. The contracts and audits explicitly test incomplete changes, wrong-object bindings, and collateral effects. For deployment, these failure modes call for appropriate authorization boundaries, action validation, and checks on affected records.

Data use. The task packages do not contain non-public user or merchant information; source materials are public or authorized for this study. Privacy transformations remove or consistently replace identifiers and sensitive values while preserving decision-relevant relations. Source-grounded synthetic records retain links to their supporting facts. Section 3.1 and Appendix A.2 describe the source routes, provenance, and separation of public execution artifacts from private authoring records.

REPRODUCIBILITY STATEMENT

All code and data for this study are available in the anonymous repository linked in the abstract. Sections 3 and 4 describe task construction, service replicas, scoring, and the evaluation protocol. Appendix A provides implementation and admission details; Appendices B and C report supplementary results and individual runs; and Appendix F records harness, judge, routing, and modality settings. Fixed initial states and archived trajectories, outputs, and verifier evidence support replay and inspection of the reported results.

REFERENCES

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*, 2024. URL https://arxiv.org/abs/2410.07095.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. TextWorld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*, 2018. URL https://arxiv.org/abs/1806.11532.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Wei Fan, Xinjie Shen, Xudong Guo, Jianhong Tu, Yang Su, Yinger Zhang, Lianghao Deng, Fengyu Wang, Baohua Dong, Yangqiu Song, and Dayiheng Liu. E-Commerce Bench: Evaluating LLM agents on long-horizon autonomous business operation. *arXiv preprint arXiv:2608.30730*, 2026. URL https://arxiv.org/abs/2608.30730.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.371/.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*, 2023. URL https://arxiv.org/abs/2310.03302.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. MetaTool benchmark for large language models: Deciding whether to use tools and which to use. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=R0c2qtalgG.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. OmniACT: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *European Conference on Computer Vision*, pp. 161–178, 2024. doi: 10.1007/978-3-031-73113-6_10. URL https://arxiv.org/abs/2402.17553.

Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, Nir Mashkif, and Segev Shlomov. ST-WebAgentBench: A benchmark for evaluating safety and trustworthiness in web agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2410.06703.

Jiaju Lin, Haoran Zhao, Aochi Zhang, Yiting Wu, Huqiuyue Ping, and Qin Chen. AgentSims: An open-source sandbox for large language model evaluation. *arXiv preprint arXiv:2308.04026*, 2023. URL https://arxiv.org/abs/2308.04026.

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. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, Albuquerque, New Mexico, apr 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. WebLINX: Real-world website navigation with multi-turn dialogue. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024. URL https://proceedings.mlr.press/v235/lu24e.html.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023. URL https://arxiv.org/abs/2305.15334.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=il5yUQsrjC.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=GEcwtMk1uA.

Qiming Shi, Yulong Tao, Linbo Jin, Zhaolu Kang, Yibo Dou, Jiawen Zhu, Tianjun Pan, Shaokang Fu, Chengyu Wang, Siyue Li, Yaping Cheng, Di Weng, and Chengfu Huo. MerchantBench: Benchmarking LLM agents for long-term coherence in e-commerce operations. *arXiv preprint arXiv:2607.28956*, 2026. URL https://arxiv.org/abs/2607.28956.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, Ye Tian, and Sujian Li. RestGPT: Connecting large language models with real-world restful apis. *arXiv preprint arXiv:2306.06624*, 2023. URL https://arxiv.org/abs/2306.06624.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. ToolAlpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023. URL https://arxiv.org/abs/2306.05301.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand, aug 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Peng Wang, Ruihan Tao, Qiguang Chen, Mengkang Hu, and Libo Qin. X-WebAgentBench: A multilingual interactive web benchmark for evaluating global agentic system. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. URL https://arxiv.org/abs/2505.15372.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*, 2022. URL https://arxiv.org/abs/2203.07540.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying LLM-based software engineering agents. *ACM Transactions on Software Engineering and Methodology*, 2025. doi: 10.1145/3715754. URL https://dl.acm.org/doi/10.1145/3715754.

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. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems 37*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks. In *Advances in Neural Information Processing Systems 38*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/0d744742f6fac4d1134c019b7cef3c8a-Abstract-Datasets_and_Benchmarks_Track.html.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems 37*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35*, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

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

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a generalist web agent, if grounded. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024. URL https://proceedings.mlr.press/v235/zheng24e.html.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL https://arxiv.org/abs/2307.13854.

Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, and Maarten Sap. SOTOPIA: Interactive evaluation for social intelligence in language agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mM7VurbA4r.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. ToolQA: A dataset for LLM question answering with external tools. In *Advances in Neural Information Processing Systems 36*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/9cb2a7495900f8b602cb10159246a016-Abstract-Datasets_and_Benchmarks.html.

APPENDIX

A CONSTRUCTION AND IMPLEMENTATION DETAILS

A.1 DESIGN PRINCIPLES

Five principles guide the design. First, authorship is accountable: every case has a human-owned workflow interpretation and outcome contract, even when an LLM assists drafting. Second, fidelity is operational and evaluated at the granularity of declared routes and behaviors through reference snapshots or goldens, interaction tests, negative paths, and state-propagation checks. Third, environments are writable: legitimate actions produce durable state transitions that later pages, reads, and verifiers observe. Fourth, difficulty comes from repeated judgment over realistic, messy evidence rather than exposed grader hints. Fifth, every score must be reproducible from a fixed start and traceable to final state, artifacts, or explicitly declared interaction evidence.

These principles keep the benchmark focused on consequential work while making provenance explicit. The agent receives an operable surface rather than a pre-digested answer table. Authors may use public real records directly or construct structured records from documented source facts, with each route labeled and its decision-relevant relations preserved. The public task is structurally separated from private seeds, ground truth, verifier logic, and mock internals.

A.2 REQUEST PROVENANCE, PRIVACY, AND ADMISSION

The transformation is substantive authoring rather than prompt paraphrase. For example, a short request for an online-store diagnosis becomes a read-only executable contract with real-URL requirements, access-block evidence, screenshot obligations, and a traceability manifest. A four-channel market-report request becomes a fixed research packet with a normalization policy, recommendation/alternative/exclusion decisions, and linked spreadsheet, CSV, JSON, and evidence contracts. These additions make the original intent executable and scoreable without revealing the rubric.

Raw request text remains private authoring material. Historical request-derived cases retain source metadata in repository history, whereas newer workflow cases more often expose task-local public-data manifests or source-fact documentation. Provenance is reported at the case level, and the suite-wide construction process is characterized as human-led and source-constrained.

Table 8: The two authoring routes and their provenance criteria. Retained request provenance identifies request-derived cases; documented procedure and source evidence identify workflow-derived cases.

Source route	**Human-owned input**	**Allowed LLM assistance**	**Provenance label**
Request-derived	Traceable request example; curator-authored workflow interpretation	Normalization, translation, drafting	Retained request provenance
Workflow-derived	Documented operating procedure plus public/authorized evidence	Draft organization and format conversion	Documented workflow provenance
Both routes	Human-owned outcome contract and admission decision	Mechanical drafting support	Human-led, LLM-assisted, source-constrained

Where privacy transformation is needed, the protocol uses a human-reviewed mapping. Direct identifiers, account information, local paths, and confidential business values are removed or consistently replaced while preserving the object graph, cross-record bindings, contradictions, temporal order, and threshold boundaries. The executable package joins an agent-visible request and optional visible inputs with private seed/truth and a verifier. The release separates public execution artifacts from author-side privacy mappings and admission records.

Table 9: Evidence-backed workspace construction and human admission. Each stage lists the human action, the evidence currently present in the public tree, and its admission criterion.

Stage	Human action	Evidence status in current tree	Admission criterion
Workflow reconstruction	Interpret goal, systems, decisions, constraints, completion	Task and review history; request metadata where retained	Provenance determines the request- or workflow-derived label
Evidence acquisition	Freeze public records or cite official facts	Concrete packets and manifests in documented pipelines	Every field, enum, and conflict follows the declared source route
Privacy transformation	Remove/replace identifiers consistently where needed	Human-reviewed author-side mapping	Preserve relations and remove sensitive release values
Case packaging	Write request, visible inputs, seed/truth, verifier	107/107 task, rubric, grader, private; 103/107 explicit workspace	Public/private allow-list and protected-path checks
Admission	Review controls and visible-only execution	Documented pipeline plus historical/targeted reviews	Report completed audit coverage

Table 10: The five service-replica forms in CommerceAgentBench. Each form carries surface-appropriate alignment evidence. Alignment covers declared task pages, commands, validations, visible outputs, and state transitions.

Replica form	Examples	Notes
Browser SPA replica	Shopify admin, Google Contacts	Captured from real login state, served offline
HTTP API / MCP mock	Gmail, Google Docs, Amazon SP-API	Token-protected state endpoints
Isolated CLI (`daemon_cli`)	Stripe, Box, Jira, Todoist, Notion	Thin wrapper, state held in a protected daemon
Replay mock	SerpAPI trends, hotels	In-memory snapshot replay
Real OSS appliance	EverShop, Oscar	Open-source store run on the container

A.3 Service-Replica Taxonomy and Alignment Evidence

Table 10 summarizes the five replica forms. CAB assesses fidelity across the declared and regression-tested task surface: the exercised pages or commands, controls, fields, validations, visible results, and state transitions.

Browser page replication, interaction, and admission. Browser pages are first-class environment artifacts in CAB. For captured-page families, an authenticated real page is acquired through the browser capture pipeline; the serialized DOM, open shadow roots, route structure, fonts, images, product CSS, and other required assets are localized so the page runs without the live service. Cross-origin embedded pages are captured separately and re-embedded locally. The result preserves the page geometry, labels, controls, accessibility/DOM semantics, and visual context that a browser agent actually observes.

Capture alone gives a static L1 replica. CAB adds an L2 interaction layer that reconnects the captured controls to a validated stateful backend. UI/session tokens distinguish legitimate page writes from bare internal-API injection; closed-set values and required fields are checked server-side. A successful edit, create, delete, merge, publish, reorder, archive, or reset changes the same durable state read by later rendered pages and the verifier. Thus the browser, API/MCP, and verifier surfaces observe one shared world rather than separate demonstrations.

Table 11: Browser replica evidence, what each layer establishes, representative current checks, and tested coverage. L1 covers page capture, L2 covers validated interaction, and the shared-state layer covers durable mutation and reset.

Browser replica evidence	**What it establishes**	**Representative current checks**	**Coverage**
Captured real-DOM page + localized assets	Task route, layout, labels, controls, open shadow DOM, fonts/images/CSS	Shopify admin/theme routes, local Polaris assets, product and discount forms; Contacts root/new/suggestions/ trash pages with no remote references	Tested pages and routes
Injected interaction adapter + validated backend	Captured controls produce the intended accepted/rejected actions	Product create/edit, theme publish/preferences, Contacts create/merge/ label/delete, invalid enum and missing-field rejection	Declared interactive controls
Shared durable state + reset	Mutation appears in later page/API reads and verifier state; seed can be restored	Cross-surface product lookup, published-theme state, trashed-contact state, token-gated verifier readout, reset after mutation	Tested state transitions and reset paths

Browser admission is route- and operation-specific. Reference snapshot or DOM assertions establish the L1 page; interaction smoke exercises the declared controls; negative tests check rejected writes and protected state access; state-propagation tests verify that later pages or readouts reflect the mutation; reset tests restore the seed. This stack checks alignment for the tested pages and actions. Page- and route-level results are the unit of fidelity analysis, combining structural, interaction, negative-path, and state-propagation evidence.

CLI alignment, isolation, and admission. High-integrity CLI mocks are admitted through behavioral verification against upstream evidence. Authors pin an upstream version, extract the benchmark-relevant command contract from source code, recursive help, and real or sandbox execution, and write a cited fact sheet covering flags, defaults, stdout/stderr, exit codes, errors, state effects, omissions, and deviations. A separate verifier compares the implementation with golden outputs derived from the real binary or independent upstream renderers and tests mutation propagation and negative paths.

At runtime the agent invokes the native command name, but the on-path executable is a thin client. It forwards argv, stdin, and cwd to a daemon running as a protected `mocksvc` user. Source and per-task state live under protected paths; benchmark tokens stay in daemon-only environment variables. The daemon executes the hidden implementation, commits mutations, and exposes token-gated `/__bench/state` and `/__bench/audit` readouts. The current registry contains seven daemon-isolated families, used by 38 of the 107 tasks. Concretely, the construction chain runs through six stages: (1) an upstream contract from source, recursive help, and real/sandbox captures at a pinned version; (2) a cited fact sheet of commands, flags, outputs, errors, state effects, exclusions, and deviations; (3) a constrained mock implementing only declared behavior with server-side validation and durable, audited mutation; (4) independent tests for golden diff, bad-input paths, state propagation, and a non-tautological oracle; (5) `daemon_cli` isolation with a thin native wrapper, `mocksvc`-owned source and state, and token-gated readout; and (6) task admission with a fresh-start seed, final-state positive/negative controls, and leakage and bypass review.

Table 12: The seven daemon-isolated CLI families, their agent-visible commands, current local regression counts (totaling 540/540), and per-family evidence coverage. Six families carry a versioned fact sheet with golden comparison; Google Workspace uses a documented tool surface plus regression smoke.

Family	Agent command	Local regression	Evidence and coverage
Todoist	`todoist`	104/104	Versioned fact sheet + real-binary/upstream-renderer golden; declared command subset
Jira	`jira`	78/78	Versioned fact sheet + golden; state-changing subset, raw JSON approximations documented
Box	`box`	83/83	Versioned fact sheet + independent renderer golden; CSV and residual error differences documented
Stripe	`stripe`	81/81	Fact sheet/captures + core-resource state machine; dynamic fields normalized
DingTalk Docs	`dws`	85/85	Fact sheet + regression golden; 21/159 upstream commands in scope, deviations documented
Notion	`ntn`	74/74	Fact sheet + golden; help, exit, OAuth, and response deviations documented
Google Workspace	`gws`	35/35	Documented Sheets/Slides tool surface + regression smoke

A.4 Recording and Controlled Growth

This construction also creates a controlled growth path: a run may produce a new recorded state, which enters a curator-operated pipeline for sanitization, seed-schema conversion, deterministic replay, request and verifier authoring, and admission review.

Recorded state supports controlled dataset growth through a curator-operated pipeline. A curator collects a post-operation snapshot, removes sensitive values, maps it into the target mock's seed schema, and replays it from a clean start. After deterministic replay succeeds, the author adds a new user request, expected state delta or artifact contract, verifier, positive control, negative controls, and leakage review. Human admission completes the transition from archived run to benchmark case. The implemented workflow combines fresh-start fixtures, durable mutation, run archives, token-gated state/audit readouts, sanitization, seed conversion, and replay, and provides a structured foundation for further automation.

A.5 Integrity and Reproducibility

Shortcut prevention. Shortcut defenses are implemented at the environment boundary rather than inferred from the agent's explanation. Closed-set fields are validated on the server, browser-facing writes require the declared session/interface path, and verifier-only readouts are separately authenticated. For high-integrity CLI mocks, the visible native-name client is separated from protected implementation and state by a daemon boundary. These controls make direct state injection harder and, when the task contract explicitly requires an interaction or audit event, let the verifier check that evidence. Rejected invalid writes or protected-endpoint probes are logged as diagnostic integrity events but do not by themselves change task reward.

Shortcut defenses are implemented and tested at build and runtime boundaries: public/private allow-list checks, protected mock source and state, a thin native CLI wrapper, capability dropping, token-gated verifier readouts, visible-only execution audits, and targeted bypass smokes. These checks show that named shortcut paths are refused or leave auditable validation events. Fidelity is supported by the environment-specific evidence chain in Section 3, and reproducibility by fresh-start fixtures plus archived trajectories, outputs, logs, and verifier evidence. Retaining the seed, image, configuration, and state digest records the initial conditions and artifacts used for replay.

Human-led case construction, environment alignment, and controlled growth

Stage I — Human workflow reconstruction. Traceable request example or curated workflow → human recovery of goals, systems, decisions, and constraints → privacy review and source-route labeling.

Stage II — Evidence-backed case and replica. Frozen public records or source-fact-grounded structured state → visible request + optional inputs + hidden outcome contract → 1:1 task-surface UI / API / CLI behavior.

Stage III — Human admission, evaluation, and archive. Valid, incomplete, and one-change controls → leakage and visible-only execution review → final-state/artifact score and auditable archive.

Controlled reuse: archived state → sanitize → seed-schema conversion → deterministic replay → request + contract + verifier → human admission.

1:1 fidelity criterion: CAB assigns one-to-one behavioral alignment at the route and operation level after snapshot/golden, interaction, negative-path, and state-propagation regression.

Figure 2: Human-led construction and controlled dataset growth. Language models may assist normalization and drafting, but human authors own workflow interpretation, evidence mapping, the outcome contract, controls, and admission. Recorded states can re-enter authoring only through a new reviewed seed-and-contract cycle.

A.6 Verifier Changes and Harness Calibration

Verifier changes are classified as agent capability gaps, task-contract gaps, or verifier misalignment and require contract-based evidence. The selected final-state validity suite uses valid, partial, surface-only, wrong-binding, collateral, alternate-path, and tolerance-boundary states; exact one-change false-collision replays provide a stricter near-correct test. Evidence coverage is reported explicitly in Section 4.4.

Harness and calibration. The same task format runs on three agent harnesses, a workstation agent, a browser agent, and a command-line agent with a browser, which lets the same task measure different frameworks as well as different models. Several controls each came from a concrete mistake. Vision capability is set explicitly, because otherwise the gateway quietly answers image turns with a different multimodal model and the substitution does not appear in the logs. Model routing is checked after launch by querying the list of served models, because the gateway falls through silently on an unknown model code rather than returning an error. Per-run timeout and action limits can be relaxed to give a slow-thinking model room, though they are left at default for the common baseline. Before a task is finalized, its difficulty is calibrated with strong, medium, and weak probes. Planted pitfalls are reviewed from the agent's point of view, so that a pitfall is objectively detectable and does not punish a defensible alternative reading, for instance a sample rather than a population standard deviation. Within a family of tasks the pitfall mechanisms are varied, so that a single mechanism does not recur across the family.

B Additional Experimental Results

B.1 Repeated-Run Summary

Table 13 collects the OpenClaw means and sample standard deviations from the main leaderboard. The underlying results for all three harnesses are in Appendix C; Section 4.2 discusses run variation and differences across harnesses.

Table 13: Multi-trial variance on OpenClaw over the 107-task base, across three independent runs per model. Pass is the binary count out of 107 as mean±sample standard deviation, and Capacity is the per-run capacity mean±sample standard deviation.

Model	Runs	Pass (mean±std)	Capacity
Claude Opus 5	3	60.3±1.5	0.898±0.006
Claude Opus 4.8	3	54.7±1.5	0.854±0.006
GPT-5.6 Sol	3	51.3±1.5	0.855±0.012
GPT-5.5	3	49.0±1.7	0.832±0.016
Claude Opus 4.7	3	48.7±1.5	0.871±0.012
Qwen 3.8 Max	3	47.3±1.5	0.810±0.006
DeepSeek V4 Pro	3	46.7±1.5	0.749±0.012
Gemini 3.6 Flash	3	46.3±1.5	0.838±0.028
DeepSeek V4 Flash	3	44.7±1.5	0.827±0.012
GLM 5.2	3	42.3±1.5	0.821±0.006
Gemini 3.5 Flash	3	39.3±1.5	0.802±0.007
GPT-5.6 Luna	3	36.3±1.5	0.783±0.015
Gemini 3 Flash	3	29.3±1.5	0.744±0.012

B.2 Results Reported by the Occamy Model Release

The Occamy release reports CommerceAgentBench under its earlier name, *RealReplicaBench*, as both an evaluation axis and an executable-environment substrate. Occamy-35B-A3B is post-trained from Qwen3.6-35B-A3B with supervised fine-tuning and mixed reinforcement learning. The release describes Data RSI as a framework that composes executable tasks from personas, tools, fixtures, skills, constraints, and graders. CAB's resettable worlds and verifiable outcomes provide an environment format suitable for collecting trajectories and execution feedback.

On the same 107-task evaluation base, the release reports a 37.3% pass rate for Occamy-35B-A3B and 19.6% for its Qwen3.6-35B-A3B base model, a difference of 17.7 percentage points. Within that release table, Occamy scores 3.7 points above DeepSeek V4 Pro and 2.0 points below GLM 5.2. These are externally reported model-release results. The thirteen-model, three-harness comparison in Table 1 is the primary evaluation in this paper.

B.3 Detailed Ablation Results

Flip counts are 12 for Claude Opus 4.8, 11 for GPT-5.5, and seven for Gemini 3 Flash (Table 3). We call a cell robust when at least two models flip, weak when exactly one flips, and no flip when none does. These are descriptive categories for cross-model agreement. Eleven of the 15 cells are robust, two are weak, and two have no binary flip. The latter two A3 cells reach best post-intervention capacities of 7/8 and 8/9. Tables 4 and 14 locate the observed outcomes by cell and family within this one-harness diagnostic panel.

The per-cell matrix (Table 4) records flips across multiple task shapes. The A3 intervention supplies target-system state while leaving source-system, side-effect, and exclusion constraints in the outcome contract. The agent must still preserve unrelated objects and produce the required deliverables. A binary flip therefore records completion of the retained contract in at least one of three attempts after the intervention.

Table 14: Flip distribution over the selected task families (flipping models per family, up to 3 per cell). The selected 15 cells cover A1 and A3 interventions across API, file, and several command-line systems.

Family	Type	#	O4.8	G5.5	Gem
API / travel	A1	1	1	1	0
File / web-data	A1	1	1	1	1
File / logistics	A1	1	1	0	0
File / commerce-ops	A1	1	1	1	0
CLI / Jira	A3	1	1	1	1
CLI / Notion	A3	1	1	1	1
CLI / Stripe	A3	1	1	1	1
CLI / Todoist	A3	1	1	1	1
CLI / xfer-customs	A3	6	3	3	2
CLI / xfer-invoice	A3	1	1	1	0

For additional localization, five variant runs are decomposed into fine-grained atoms and checked against the official grade. In the two examples reported here, `lc-eco-tote` places all 14 items correctly but misses three rule citations, while `cargo-insurance` passes all 20 per-customer amount/status atoms and all six summary atoms but reaches 10/15 on reason-tier atoms. These examples show how per-check evidence distinguishes completed execution from a remaining closing judgment and motivate a larger failure-distribution study.

Table 15 reports the complete per-cell check counts and best completion fractions, complementing the main-paper flip matrix.

Table 15: Per-cell ablation outcomes. Type is the ablated burden (A1 judgment, A3 state); Checks is the task's check count; Flips is the number of distinct models (of three) that flip fail-to-pass once the burden is removed; Best is the highest check fraction any model reached. The two no-flip cells reach 7/8 and 8/9 checks while retaining an unmet requirement.

Cell	Type	Checks	Flips	Best	Tier
api-travel-asia-supplier-tour-handoff	A1	12	2	12/12	robust
file-google-trends-data-quality-audit	A1	5	3	5/5	robust
file-logistics-cargo-insurance-alloc.	A1	7	1	7/7	weak
cli-jira-supplier-defect-triage	A3	15	3	15/15	robust
cli-notion-supplier-evidence-upload	A3	7	3	7/7	robust
cli-stripe-billing-audit-repair	A3	13	3	13/13	robust
cli-todoist-sourcing-schedule	A3	75	3	75/75	robust
cli-customs-dual-source-reconcile	A3	8	1	8/8	weak
cli-customs-export-control-relabel	A3	8	0	7/8	no flip
cli-customs-medtech-quality-dossier	A3	10	2	10/10	robust
cli-customs-tiered-approval-routing	A3	8	3	8/8	robust
cli-customs-trade-finance-pipeline	A3	9	0	8/9	no flip
cli-customs-zibo-evidence-room	A3	10	2	10/10	robust
cli-xfer-invoice-followup-queue	A3	12	2	12/12	robust
file-lc-eco-tote-discrepancy-audit	A1	17	2	17/17	robust

B.4 Difficulty Distribution and Discrimination

Beyond the ranking, we examine how cleanly the suite separates models. The three-harness average spans 59.7% for Claude Opus 5 to 29.2% for Gemini 3 Flash, so the suite neither saturates nor bottoms out at the aggregate level. The top two systems are separated from the middle, while several

adjacent models in the middle differ by only one or two percentage points and are best interpreted together with task-level uncertainty.

Binary pass and capacity broadly track the same ability gradient but disagree on several middle-model comparisons, indicating that closing all required decisions and accumulating partial capability are related but distinct signals. All completed cells in the main tables cover the same 107-case evaluation set; its modality composition is described in Appendix D.

B.5 Scoring Audit Notes

Section 4.4 reports the state matrix, readout layers, and one-change replay results. Mock-backed deterministic checks read committed business state; other deterministic checks read files or structured artifacts. Event and audit gates are part of the corresponding task contract. The audit covers 14 verifiers, and the exact one-change replay covers 12 of 32 HARD cases. The 35 model-judge checks use grounded boolean propositions in six cases; the reported state and replay audits concern the selected verifier controls.

C Per-Run Results

Table 1 reports each harness cell as a mean over three independent runs. Tables 16 and 17 give the underlying binary pass counts and capacity scores for all thirteen models on Accio, Pi, and OpenClaw. Every run covers the full 107-case base.

Table 16: Per-run binary pass counts (out of 107) and capacity scores behind the means in Table 1, part 1 of 2. R1–R3 are three independent runs per cell.

		Binary pass (/107)			**Capacity**		
Model	**Harness**	**R1**	**R2**	**R3**	**R1**	**R2**	**R3**
Claude Opus 5	Accio	66	68	65	0.861	0.896	0.872
	Pi	65	64	66	0.905	0.897	0.900
	OpenClaw	60	62	59	0.905	0.893	0.896
Claude Opus 4.8	Accio	59	61	58	0.886	0.898	0.875
	Pi	56	58	55	0.895	0.898	0.906
	OpenClaw	55	53	56	0.860	0.848	0.855
Qwen 3.8 Max	Accio	56	54	58	0.870	0.878	0.885
	Pi	53	55	52	0.856	0.867	0.845
	OpenClaw	47	49	46	0.804	0.816	0.810
DeepSeek V4 Pro	Accio	54	56	53	0.793	0.853	0.821
	Pi	53	51	54	0.902	0.914	0.879
	OpenClaw	47	45	48	0.749	0.737	0.761
GPT-5.6 Sol	Accio	55	57	54	0.873	0.896	0.884
	Pi	52	55	55	0.895	0.907	0.875
	OpenClaw	53	50	51	0.855	0.867	0.843
Claude Opus 4.7	Accio	56	58	55	0.878	0.890	0.867
	Pi	52	54	51	0.867	0.855	0.877
	OpenClaw	49	47	50	0.871	0.859	0.883
GLM 5.2	Accio	50	52	51	0.787	0.795	0.807
	Pi	50	49	51	0.804	0.795	0.796
	OpenClaw	42	44	41	0.814	0.823	0.826

D Run Statistics and Modality Labels

Tables 16 and 17 report the three aggregate run scores behind each model–harness mean and sample standard deviation. The leaderboard uses the full 107-case evaluation set. The intervention panel uses a different statistic: a cell–model pair flips when at least one of its three intervention runs passes the full contract. The per-cell matrix therefore records observed success within three attempts.

The evaluation set contains 67 text-only, 20 vision-required, and 20 browser-text-capable cases. These labels describe the required input and interaction form. All completed main-table cells cover the 107-case set under their declared model–harness configuration, including the required visual input. A text-only configuration evaluated on 67 cases uses a separate denominator. The browser-interface count and CLI-family count describe interface use, which can overlap with these modality labels.

E Case Studies

The following cases connect reported scores to concrete workflow requirements. Per-check outcomes identify unmet contract items; trajectories, logs, and screenshots provide the corresponding execution evidence. The cases cover billing repair, schedule coordination, mail triage, and browser booking, with both complete and partial outcomes.

Table 17: Per-run binary pass counts and capacity scores, part 2 of 2. Together, Tables 16 and 17 report all thirteen models and all three harnesses.

		Binary pass (/107)			**Capacity**		
Model	**Harness**	**R1**	**R2**	**R3**	**R1**	**R2**	**R3**
GPT-5.5	Accio	48	51	51	0.864	0.876	0.834
	Pi	47	50	50	0.884	0.823	0.876
	OpenClaw	51	48	48	0.835	0.815	0.847
Gemini 3.6 Flash	Accio	50	52	51	0.815	0.827	0.796
	Pi	47	50	50	0.892	0.850	0.869
	OpenClaw	48	45	46	0.867	0.834	0.812
DeepSeek V4 Flash	Accio	50	52	49	0.838	0.883	0.850
	Pi	46	49	48	0.861	0.873	0.811
	OpenClaw	46	43	45	0.827	0.839	0.815
Gemini 3.5 Flash	Accio	46	48	45	0.821	0.833	0.819
	Pi	44	42	45	0.847	0.859	0.835
	OpenClaw	39	41	38	0.798	0.798	0.810
GPT-5.6 Luna	Accio	48	46	50	0.809	0.797	0.809
	Pi	42	44	41	0.807	0.810	0.819
	OpenClaw	36	38	35	0.797	0.785	0.767
Gemini 3 Flash	Accio	31	34	33	0.769	0.759	0.757
	Pi	31	32	32	0.796	0.808	0.756
	OpenClaw	31	28	29	0.744	0.756	0.732

Billing repair. On `cli-stripe-billing-audit-repair` (13 checks), no evaluated model passes the binary gate in the reported OpenClaw case study, while mean capacity is 0.87. The A3 variant supplies target-system state, and all three ablation models achieve 13/13 in at least one attempt. The original task requires dependent billing updates to satisfy the final contract. The two measurements record partial completion of the original workflow and full completion after the state intervention.

Schedule coordination. `cli-todoist-sourcing-schedule` carries 75 checks and has no complete workflow in the reported case study, with mean capacity 0.83. Each of the three ablation models reaches 75/75 in at least one A3 attempt. Every required batch outcome contributes a check within this task; macro-averaging assigns the task the same benchmark weight as a task with fewer checks. Its scores describe completion under that declared schedule rubric.

Mail triage. `api-gmail-supplier-rfq-triage` has 23 checks. In one OpenClaw run, Claude Opus 4.7 passes 23/23, Gemini 3 Flash reaches 22/23, GPT-5.5 reaches 20/23, and Claude Opus 4.8 and Gemini 3.5 Flash each score 0/23. The two partial runs leave one and three checks unmet, respectively. Capacity distinguishes these outcomes from zero completed checks, while the binary verdict records completion of the entire request.

Browser booking. The `browser-freightos-booking` family, including rated and fastest variants, reaches full capacity for every model tested on it despite carrying 28–31 checks per task. The workflow requires several interface steps with few open-ended decisions. Alongside the incomplete billing and scheduling cases, this result shows that a larger check count can accompany either a fully completed or a partially completed workflow.

F Experimental Configuration

Harnesses. The reported experiments use Accio, Pi, and OpenClaw. Each harness drives the browser, command line, API, and file surfaces under its own agent loop while receiving the same task

package, mock environment, and verifier for a case. The comparison holds the task package, initial environment, and verifier fixed across harnesses; each harness retains its own execution loop.

Judge and scoring. The evaluation-set rubrics declare 834 checks: 799 non-judge and 35 grounded boolean judge checks. Judge calls contribute only to six cases; the other 101 cases are deterministic-only. For case i, `passed` is true only if the grader completes, m_i is nonzero, and all required g_{ij} checks pass. Capacity is $(1/m_i)\sum_j g_{ij}$, macro-averaged over cases. Mock-backed and other deterministic checks read final environment state or artifacts, not the agent's claim. Check granularity follows the capability-unit policy in Section 3.4. The selected state audit covers 14 verifiers, and exact one-change replay covers 12/32 HARD cases. All model-judge checks use `gemini-3.1-pro-preview`; deterministic checks are evaluated by their corresponding verifier code.

Routing and modality. Runs use parallelism two, with per-task timeouts of about 1800 seconds and 1500 seconds for browser subagents. Thinking effort is set per model and recorded with the run configuration. Vision capability is set explicitly per model and routing is verified after launch by querying the served model list, because the gateway otherwise falls through silently on an unknown model code and can answer image turns with a different multimodal model without surfacing the substitution.

Evaluation base. All results are on the 107-case evaluation base, an ILP-trimmed subset of the full suite chosen to balance modality and domain coverage while removing redundant near-duplicate tasks. The base divides into 67 text-only, 20 vision, and 20 browser-text-capable cases.