WHERE DOCUMENTATION ENDS: SOURCE-DERIVED BEHAVIORAL SPECIFICATIONS FOR LLM TOOL SIMULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A common way to evaluate, and increasingly to train, LLM agents is to replace the tool backends in their sandboxes with an LLM that simulates each tool response, because real backends are costly, unstable, or unsafe to call at scale. These simulators are usually conditioned on API documentation, while real backends also enforce rules that no document states. We study this documentation gap on three author-built tool servers with 38 frozen hidden rules, 31 of which never fire in any demonstration. Doc-only failures concentrate on rules that contradict a documentation-shaped prior (17.2% vs. 69.6% accuracy), reference-free plausibility judges do not detect them (Youden $J \leq 0.04$), and they change what agents do under real-backend replay. Comparing channels for teaching a simulator the never-triggered rules, specifications derived offline from source reach 80.3% first-encounter accuracy, against 22.7% for documentation and 59.1% for 64-call active probing; a generic summarization prompt does as well, so the gain comes from the channel, not the compilation method. Downstream, source-derived specifications repaired both focal doc-only failures, with success depending on whether a rule fired rather than on its exact value. On three public MCP servers, undocumented behavior, judge blindness, and false-success outcome changes all recur.

1 INTRODUCTION

Agent pipelines increasingly replace real tool backends with language-model simulators: in evaluation (Ruan et al., 2024; Guo et al., 2024; Zhou et al., 2025), in training-data synthesis (Tang et al., 2023; Liu et al., 2025b; Kimi Team, 2025; Castellani et al., 2025), and as RL environments (Li et al., 2025; Ren et al., 2025; Zhang et al., 2026; Zuo et al., 2026). Real backends are costly, unstable, or unsafe to call at scale; a simulator is cheap but only as faithful as the information it conditions on, and in these pipelines that information is mainly documentation: tool specifications (Ruan et al., 2024), API documentation plus a few cached real responses (Guo et al., 2024), OpenAPI specifications (Tang et al., 2023), or tool schemas (Zhang et al., 2026). Software engineering has long documented that documentation and code diverge. Static checks find over 1,000 documentation errors in five widely used libraries (Zhong & Su, 2013) and 1,158 defective directives among about 2,000 JDK API constraints (Zhou et al., 2017); 19% of studied Android crashes might have been avoided had the thrown exceptions been documented (Kechagia & Spinellis, 2014); developers rank incompleteness and incorrectness among the severest documentation problems (Uddin & Robillard, 2015); and 97.1% of 856 MCP tool descriptions show at least one quality smell (Hasan et al., 2026). Real services therefore enforce behavior that their documentation omits: quantity caps, silent value transformations, refusals on particular state combinations. An enterprise benchmark embeds thousands of business rules (Gupta et al., 2026), and Qwen-AgentWorld reports factuality as the weakest dimension of its simulator (Zuo et al., 2026).

A doc-only simulator cannot know a rule that is absent from its documentation. This follows by construction; what it leaves open are three questions. **(Q1) Locus and visibility.** Which undocumented rules does a doc-only simulator get wrong, and do the reference-free checks used in simulator pipelines detect the errors? **(Q2) Channels.** If some extra information is available offline, which kind teaches a simulator rules that were never observed in any interaction log? **(Q3) Con-**

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.

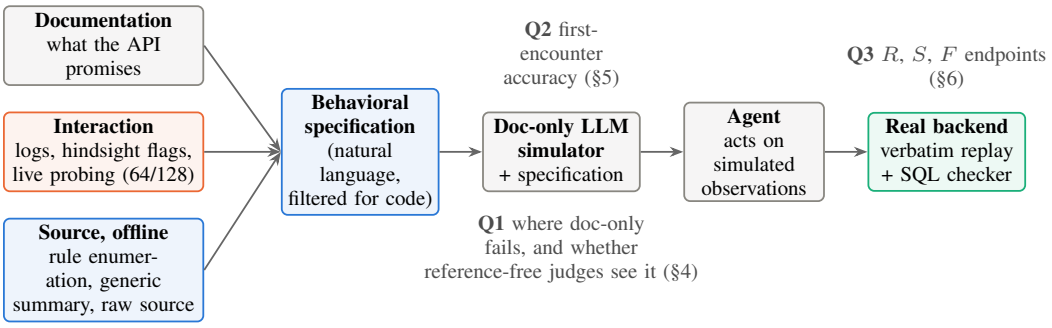

Figure 1: Study overview. A doc-only simulator is given one of several natural-language artifacts, each derived from a different information channel. We ask where doc-only simulation fails and whether judges notice (Q1), which channel teaches rules that never fired in any demonstration (Q2), and whether the difference changes agent outcomes under real-backend replay (Q3).

sequence. Do fidelity gains on single responses change what happens downstream, when an agent acts on simulated observations and is scored against a real backend?

We answer these questions on three SQLite tool servers with 38 hidden rules. We build our own servers because the questions need what real servers cannot give: a complete, known rule list, control over which rules the demonstrations trigger, and source no model has seen; we then check the resulting phenomena on three public MCP servers (§7). The rule inventory, demonstration seeds, and never-triggered designations were frozen by hash before any teaching arm ran. Every accuracy number is an exact canonical-JSON match against real execution; no LLM judge enters any primary metric.

Contributions.

- **An anatomy of the documentation gap (Q1, §4).** Doc-only failures concentrate on rules that contradict what the documentation leads a model to expect: 17.2% accuracy on those rules against 69.6% on rules that extend the prior. Failures persist across the model tiers we tested, and state-blind plausibility judges accept 540 of 588 hidden-rule errors ($J \leq 0.04$). History-aware judges trade missed detections for false-positive rates of 45 to 85%, while reference-based metrics separate the same errors perfectly.
- **An information-channel comparison (Q2, §5).** On the 31 never-triggered rules, specifications derived from source beat every interaction-derived channel we built at the budgets we tested, and the ordering holds with a second student; on public MCP servers the benefit requires raw source (§7). Raw-source prompting, a rule-enumeration prompt, and a generic "write the documentation" prompt land within 1.6 points of each other, against a gold-description bound of 87.9%. An audit of every specification item shows that probing-derived cards also state wrong rules at a much higher rate.
- **A preregistered downstream test (Q3, §6).** Under a plan frozen before the first episode, source-derived specifications repaired both focal doc-only failures with zero false completions. Downstream success needs only a signal that a rule fired, so exact match can understate a specification's value; conversely, a wrong probing-derived rule broke tasks the documentation-only simulator had passed.

2 Related Work

Simulated tool environments. Beyond the prompted simulators cited in §1, MirrorAPI trains a simulator on real request–response pairs from thousands of APIs (Guo et al., 2025). AgentBench, WebArena, τ-bench, and MCP-Bench keep a real backend (Liu et al., 2024; Zhou et al., 2024; Yao et al., 2024; Wang et al., 2025). EnvSimBench benchmarks environment simulators and shows that plausible surface form can mask state errors; its code-conditioned setting corresponds to our raw-source arm (Liu et al., 2026b). MIRAGE conditions an LLM dependency simulator on white-box

source for microservice testing and reports that source alone is often enough (Zhang, 2026); we study which channel teaches rules that never fired, and what the resulting errors do to agents.

Judging without references. LLM judges have known biases (Zheng et al., 2023; Wang et al., 2024a; Gu et al., 2024). Heuristic judges of tool-using agents can agree with humans only at chance level (Gurram, 2026), and behavior-level consistency diverges from surface similarity in text world models (Huang et al., 2026). Our audit isolates one structural case: hidden-rule errors are schema-valid, so a judge whose evidence is the documentation has nothing to flag.

Where environment knowledge comes from. Experience-driven agents distill guidelines or manuals from their own rollouts (ExpeL, AutoManual; Zhao et al., 2024; Chen et al., 2024), learn from tool-use traces (Schick et al., 2023; Yao et al., 2023), or learn from failures (Wang et al., 2024b). Privileged-information distillation conditions a teacher on reference trajectories, answers, runtime state, or patches (Penaloza et al., 2026; Choudhury & Sodhi, 2024; Ma et al., 2026; Chen et al., 2020). Policy internalization compiles *written* rules into models (Liu et al., 2025a; Yang et al., 2025). Documentation-grounded tool use assumes the documentation is complete (Patil et al., 2024). Offline API self-exploration and learning from API feedback are probing-channel methods (Liu et al., 2026a; Polyakov et al., 2025). Specification synthesis turns documentation or code into formal specifications for verification (Ma et al., 2025; Xia et al., 2026), and CodeMidas builds RL environments from codebases (Ye et al., 2026); we instead compile source into natural-language cards for a simulator. These channels teach a rule only after it has been observed or written; source states rules that no rollout has exercised. We compare channels, not methods; stronger exploration is untested (§8).

3 Testbed and Protocol

Servers and rules. Three single-file Python servers backed by SQLite expose documented operations: *inventory* (16 hidden rules), *booking* (16), and *projecthub* (6). Each server also implements hidden rules that appear in source but not in documentation. The rules fall into three classes: 17 refusals (ERROR), 16 silent transformations such as clamps and surcharges that return success-shaped responses (SILENT), and 5 derived-state rules (DERIVED). Per-rule detector predicates label, for every real step, which rules fired. The inventory and the teacher-demonstration seeds were frozen by SHA-256 hash. Under those seeds, **31 rules never fire**; we call them never-triggered (NT). The *strict-29* cohort drops two NT report rules whose effects can surface indirectly in demonstrations.

Teacher-forced fidelity. Given the documentation, the real interaction prefix, and the next call, the simulator must emit the exact response; because the prefix is real, errors do not compound. The primary channel endpoint is *first-encounter* accuracy: the step at which a trajectory first reaches a given NT rule, scored by exact canonical-JSON match. Evaluation uses scripted trajectories on seeds disjoint from the demonstrations: 66 first-encounter observations over 31 rules, plus 114 non-rule steps as a regression check.

Unit of inference. Observations cluster within rules, so the rule is our unit. For paired arm contrasts we compute each rule's accuracy under both arms and take the mean paired difference. We report an exact one-sided sign-flip test over rules with a nonzero difference, and a 95% rule-bootstrap interval (10k resamples). The inferential target is this frozen inventory, not a population of servers; observation-level Wilson intervals are descriptive only.

Models. The simulator is Claude Haiku 4.5 in all channel experiments, and specification writers use Claude Sonnet 5; using a weaker student than teacher means that information, not simulator capability, carries the effect. One arm uses a Kimi teacher to check teacher-family sensitivity. The gap audit (§4) adds Sonnet- and Opus-tier simulators; the E1 agent is Claude Sonnet 5. The student runs with thinking disabled, 400 output tokens, and default temperature; Sonnet teachers use adaptive thinking at medium effort. Model sampling is not seed-controlled.

Table 1: The documentation gap, as measured on trigger-rich audit corpora (doc-only simulators). All entries are counts over raw per-item records. Tiers are measured on partly different rule pools (see text), so the tier row is not a scaling curve.

Measurement	Result	Notes
Documented steps, by tier (Haiku/Sonnet/Opus)	99.1 / 97.7 / 100%	near ceiling
Hidden-rule steps, by tier	55.8 / 61.9 / 66.4%	n=43 / 113 / 113
SILENT rules (Sonnet, 16 rules)	40/158 (25.3%)	12 rules at 0/n
Prior-compatible vs. prior-contradicting	71/102 (69.6%) vs. 124/720 (17.2%)	5 vs. 33 rules
State-blind judge J (Haiku / Sonnet / 2 prompt variants)	0.038 / −0.004 / 0.027	540/588 errors "plausible"
History-aware judge (Haiku / Sonnet, 300-item slice)	J=0.207 / 0.053	FPR 45.3% / 84.7%
Reference-based (exact, rubric, rule oracle)	J=1.0	needs ground truth

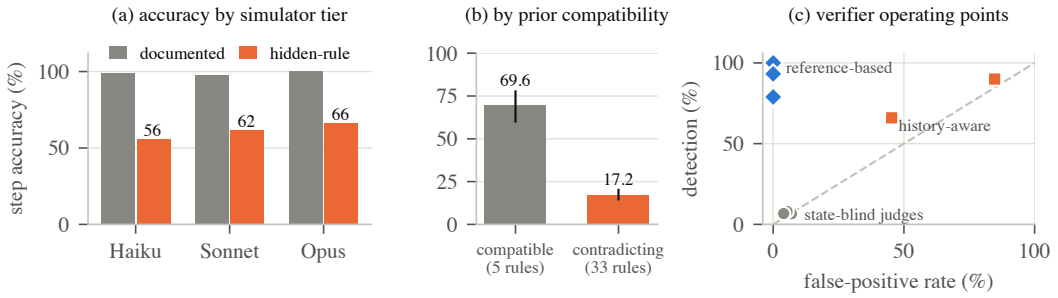

Figure 2: Anatomy of the documentation gap. (a) Doc-only simulators stay near ceiling on documented steps at every tier but not on hidden-rule steps (tiers use partly different rule pools). (b) Accuracy on hidden-rule steps by blind-panel prior class (Wilson 95% intervals). (c) Operating points of the verifiers we audited on 588 hidden-rule errors and 300 correct controls: state-blind judges sit on the chance diagonal, history-aware judges move along it, and only reference-based metrics (exact match, rubric, rule oracle, surface similarity) leave it.

4 Anatomy of the Documentation Gap

Where the errors are. Table 1 summarizes the audit. On documented behavior, doc-only simulators sit at or near ceiling. On hidden-rule steps they reach 55.8 to 66.4%, and on silent transformations only 25.3%. Twelve of the sixteen silent rules are never simulated correctly. Nine of those twelve carry an arbitrary constant, such as a cap of 1000 or a floor price; the other three are guards or no-ops. The tier row mixes instruments: the Haiku value comes from an earlier harness, while the Sonnet and Opus values share a legacy pool. We therefore read it only as persistence across tiers, not as a scaling trend. A blind five-vote LLM panel, shown only the documentation and each rule's description, labeled each rule as compatible with, or contradicting, the prior that the documentation induces. Accuracy on the 33 prior-contradicting rules is 17.2%, against 69.6% on the 5 compatible ones. The compatible group is small, and the panel may share the simulators' priors. All 38 labels were unanimous, but the five votes come from one model, so unanimity shows low sampling variance, not independent agreement; we did not calibrate against human labels. We read the split as a diagnostic of which rules to worry about, not as a causal estimate.

What deployed checks see. We scored 588 hidden-rule errors and 300 correct controls with the metric families used in simulator pipelines. J is the flag rate on errors minus the flag rate on controls. Each judge makes one YES/NO plausibility call per item (NO counts as a flag), the rubric flags scores of 2 or less on a 1–5 scale, and the controls are a fixed stratified sample; all of this was preregistered. A state-blind plausibility judge, which sees the documentation, the call, and the response, calls 540 of the 588 errors plausible. Its J is at most 0.04 at both tiers and under two alternative prompts. Giving the judge the full real history does not produce a usable operating point: the mid-tier judge flags 45.3% of correct controls, and the stronger judge 84.7% (J=0.053). Exact match, a reference-based rubric, and a deterministic rule oracle all reach J=1.0. The errors are schema-valid, so documentation and history do not determine the hidden values; paired ground truth does (for the judge configurations we tested).

Table 2: First-encounter accuracy on never-triggered rules (%). NT-31: 66 observations over 31 rules (Wilson 95% CI). Rule mean: unweighted mean over rules. Non-rule: 114 regression steps. All arms use the same Haiku 4.5 student.

Channel	Arm	NT-31 [95% CI]	strict-29	rule mean	non-rule
Documentation	docs only	22.7 [14.3, 34.2]	19.0	21.0	98.2
	docs-notes	19.7 [11.9, 30.8]	–	15.3	91.2
Interaction	passive logs K=2 / 8 / 32	28.8 / 27.3 / 25.8	15.5[a]	24.2 / 22.6 / 21.0	97.4[a]
	trajectory-PI (K=32)	22.7 [14.3, 34.2]	–	17.7	97.4
	active probing, 64/domain	59.1 [47.0, 70.1]	60.3	59.7	90.4
	active probing, 128/domain	54.5 [42.6, 66.0]	50.0	52.4	93.9
Source	rule enumeration (SIBYL)	80.3 [69.2, 88.1]	77.6	79.0	95.6
	same, Kimi teacher	78.8 [67.5, 86.9]	77.6	78.2	97.4
	generic summary	81.8 [70.9, 89.3]	79.3	80.6	94.7
	raw source in prompt	80.3 [69.2, 88.1]	77.6	79.0	94.7
Reference	gold rule descriptions	87.9 [77.9, 93.7]	86.2	87.1	94.7

[a] K=32 only.

What the errors do. We ran each agent task twice: once with observations from the real backend, and once with observations from the doc-only simulator. In both conditions, the agent's write calls are replayed verbatim on a fresh real backend and checked by a deterministic SQL predicate. The manipulated variable is therefore only where the agent's observations come from. On recovery-contingent tasks, whose goals contain an explicit "if refused, do X" branch, 121/160 pairs change outcome (three rules change 40/40, one 1/40). On end-state-only tasks, 9/40 change outcome; eight are on the receive-cap and price-jump tasks (4/4 each), real-pass and simulator-fail, and one runs the other way. In the dominant direction, the simulator returns a success-shaped response where the real server clamps or refuses, so the agent never takes the recovery step and often declares completion. We call this *behavioral degradation with false completion*. It is not a simulator-side grading error, which we measure separately in §6. Because a handful of rules drive these counts, we do not attach population-level intervals to them. The end-state tasks became the focal tasks of E1.

5 Which Channel Teaches Never-Triggered Rules?

Arms. All arms share the doc-only student, prompt template, and evaluation plan, and differ only in a natural-language artifact prepended to the prompt (or the raw source itself).

- *Interaction-derived arms.* Passive logs: the teacher writes notes from $K \in \{2, 8, 32\}$ real transcripts. Trajectory-PI: K=32 transcripts with every rule-governed step flagged in hindsight. Active probing: the teacher operates a live sandbox with 64 or 128 calls per domain under an explicit boundary-testing brief, then writes notes from its probe log.
- *Documentation arm.* Docs-notes: the teacher writes notes from the documentation alone.
- *Source-derived arms.* Rule enumeration (*SIBYL*): the teacher reads comment-stripped source and enumerates undocumented behaviors, each with a contrastive example (call, documentation-predicted response, actual response). Generic summary: the same teacher and source with a plain prompt, "write complete developer documentation for this server's tools", with no mention of hidden rules and no examples. Raw source: the student receives the source itself. Gold: numbered behavioral descriptions copied from the frozen rule inventory.

All teacher-written artifacts pass a contamination filter that rejects source-only identifiers and 5-grams shared with the source (documented API surface and response literals are allowed); offending cards are regenerated.

Logs do not contain what never fired. Growing passive logs sixteen-fold barely moves NT accuracy (28.8% to 25.8%, Table 2), and hindsight flags do not help (22.7%); none of the strict-29 rules appears in any transcript. The same passive teachers are accurate on the hidden rules that *did* appear

Table 3: Rule-level paired contrasts (mean per-rule difference in points; exact one-sided sign-flip over rules; 95% rule bootstrap).

	31 NT rules			strict 29		
Contrast	Δ	p	95% CI	Δ	p	95% CI
SIBYL − docs	+58.1	<.001	[38.7, 75.8]	+58.6	<.001	[37.9, 77.6]
active-64 − docs	+38.7	<.001	[19.4, 58.1]	+41.4	<.001	[22.4, 60.3]
SIBYL − active-64	+19.4	.024	[1.6, 36.3]	+17.2	.056	[−1.7, 34.5]
generic summary − active-64	+21.0	.012	[5.6, 37.1]	+19.0	.030	[1.7, 36.2]
generic summary − SIBYL	+1.6	.50	[−12.9, 16.1]	+1.7	.50	[−13.8, 17.2]
raw source − SIBYL	0.0	.59	[−11.3, 12.9]	0.0	.59	[−13.8, 13.8]
gold − SIBYL	+8.1	.14	[−1.6, 19.4]	+8.6	.14	[−1.7, 20.7]

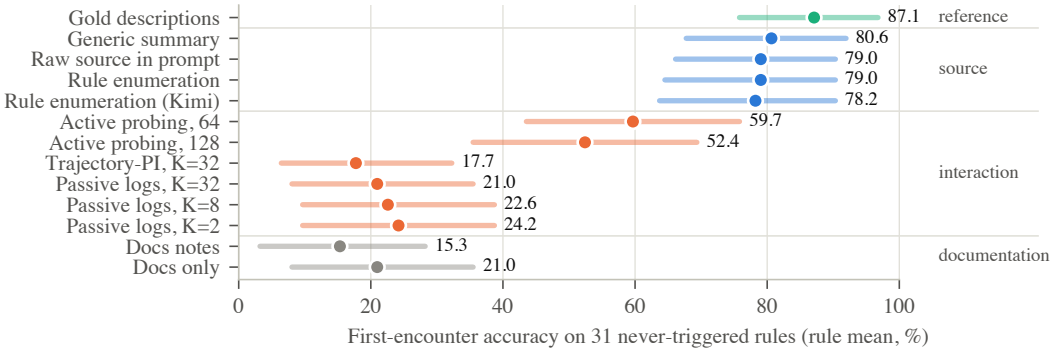

Figure 3: Channel comparison at the rule level. Dots: mean first-encounter accuracy over the 31 never-triggered rules; bars: 95% rule-bootstrap intervals. All source-derived artifacts cluster together regardless of how the source was compiled; passive interaction does not move off the documentation baseline.

in their logs: 14/17 at K=8 and K=32, against 2/17 for documentation alone. They teach what they see; documentation-only notes are slightly worse than none (19.7%).

Probing discovers, but at the tested budgets it trails source. Active probing is the strongest interaction channel: 59.1% at 64 calls per domain, and 54.5% at 128. Against it, source-derived specifications gain +19.4 points per rule (sign-flip p=.024). The gap narrows on strict-29 (+17.2, p=.056, CI includes 0). We therefore call it a substantial observed advantage at the tested budgets, not a decisive separation. The strengths are complementary at the rule level: the enumeration card wins on the receive-cap clamp, which neither probing card discovered. probing wins on the price-jump cap, which the enumeration card described but the student misapplied. E1 was designed around this crossover.

The compilation method does not matter much; the channel does. The generic "write the documentation" prompt has no rule enumeration, no mention that the documentation is incomplete, and no contrastive examples, yet it matches the enumeration prompt (+1.6 points, p=.50). Both match giving the student the raw source (Δ=0), and a Kimi teacher gives the same result. What carries the gain is that the teacher reads the source, not how it is asked to summarize it.

Recall versus application. Gold rule descriptions have perfect recall and still reach only 87.9%; the residual is *application* error (right rule, wrong exact value), mostly on composed booking-price adjustments. Better recall could buy at most the 7.6-point gap between gold and the enumeration card (+8.1 per rule, p=.14).

Probing-derived specifications contain wrong rules. First-encounter accuracy is blind to false rules that a card asserts about behavior the student would otherwise simulate correctly, so we audited every numbered item in the two cards used downstream against the source (Figure 5). Source-

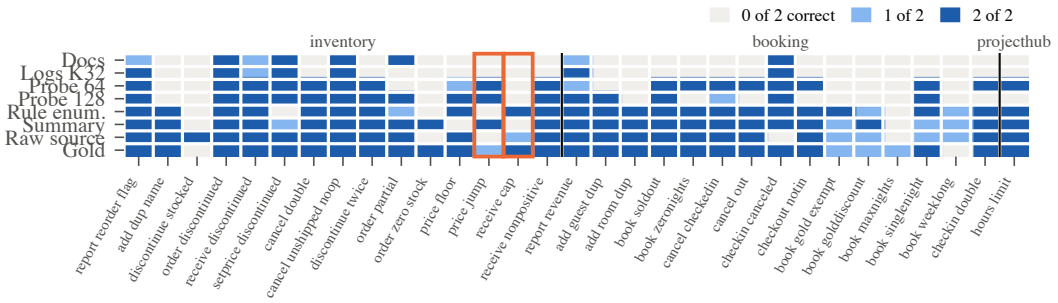

Figure 4: Per-rule first-encounter outcomes (columns: never-triggered rules grouped by server; rows: arms). Outlined columns mark receive-cap, which the enumeration card teaches and both probing cards miss, and price-jump, the reverse; these two rules became the focal tasks of E1. The generic summary shows the opposite pattern to the enumeration card on both, so rule-level strengths vary within the source channel as well. Full counts are in Table 5.

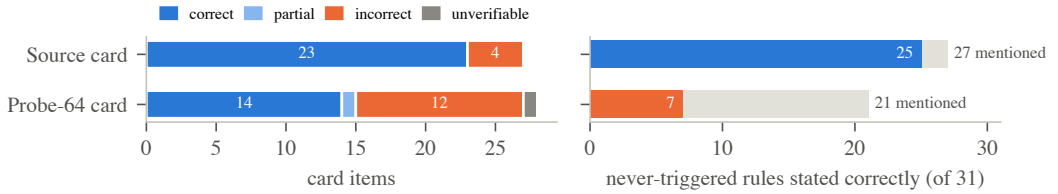

Figure 5: Item-level audit of the two specification cards used in E1. Left: each numbered card item classified against the server source as correct, partially correct, incorrect or over-general, or unverifiable. Right: never-triggered rules stated by at least one correct item (colored), and rules mentioned by any item, correct or not (gray).

derived items are correct 85% of the time; the four exceptions (three projecthub items, one booking item) over-generalize a correct rule, e.g. asserting that repeat cancellation fails. Probing-derived items are correct 50% of the time. Its incorrect items typically generalize from a few probes to a mechanism the server lacks: the booking card states that "room capacity limits how many nights can be 'in play' … for that room at once"; the server in fact refuses a non-gold booking once a room has three active reservations, regardless of capacity or duration. The same card compresses the three pricing rules into one wrong formula. The audit also shows that the probing card's 59.1% partly reflects NT errors the student guesses correctly without any card: it strictly states only 7 of the 31 rules. Probing cards also lower non-rule accuracy to 90.4% (Table 2); §6 shows the downstream cost.

Robustness. Paraphrasing the evaluation scenarios keeps most of the gain (44/60 correct with source-derived cards, 15/60 without), and a card listing only rule *names* gives none (15/66): content, not vocabulary, carries the effect. The contamination filter is a heuristic gate, not a proof that nothing leaks, and the comparison is information-asymmetric by design: the source arms read a complete rule statement, while the probing arm has to find the rules. The finding is that, at our budgets and without normalizing teacher effort (Table 9), the source channel delivered more correct rules and, in the two cards we audited, fewer wrong ones; not that exploration cannot in principle find these rules.

Replications (Appendix B). With DeepSeek-V3 as the student on the same frozen artifacts, the ordering holds: source arms reach 73.4 to 80.6 rule mean, active-64 66.9, and docs 20.2. The source-over-probing gap shrinks, however, and is no longer significant (SIBYL − active-64: +13.7, p=.076). Two fresh artifact draws per arm leave SIBYL stable (79.0 to 82.3 rule mean) but spread the generic summary (64.5 to 85.5), and the original active-64 card was its best draw (43.5 to 59.7);

Table 4: E1 real-replay success R (160 episodes, 0 parse failures, 0 invalid terminal reads). Tasks not shown pass in every condition. Focal tasks: 8 replicates; others: 2.

Task	real	docs-sim	A64-sim	SIBYL-sim
inv_hr_cap (focal)	8/8	0/8	0/8	8/8
inv_hr_price (focal)	8/8	0/8	8/8	8/8
bk_hr_soldout	2/2	2/2	0/2	2/2
bk_hr_cancel	2/2	2/2	0/2	2/2
bk_hr_checkout	2/2	2/2	1/2	2/2
bk_hr_single*	2/2	2/2	1/2	2/2
controls (4 tasks)	8/8	8/8	5/8	8/8
H1, cap: SIBYL > A64	one-sided Fisher p=7.8×10^{-5} (passes)			
H1, price: A64 > SIBYL	one-sided Fisher p=1.0 (fails); H1 not established			
False completion $P(F{=}1, R{=}0)$, primary	0/28	16/28	8/28	0/28
Simulator grading $P(S{=}1, R{=}0)$	–	16/24	8/24	0/24

*Goal discloses hidden-rule outputs; excluded from the primary cohort.

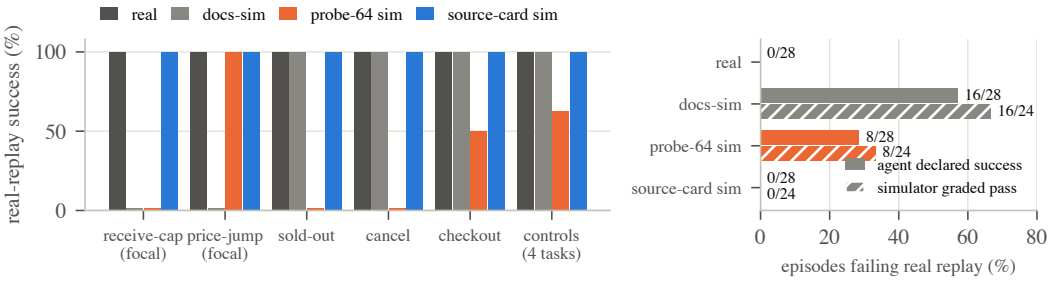

Figure 6: E1 outcomes. Left: real-replay success by task and observation source (tasks where every condition passes are omitted). Right: among primary-cohort episodes, the share in which real replay failed while the agent declared success (solid) or the simulator's own terminal reads graded the episode as passed (hatched).

averaged over draws, SIBYL − active-64 is +30.6 (p<.001). Graded scores keep SIBYL ahead of active-64 (p=.035, .017), but a server-blocked bootstrap gives [−5.9, 41.1].

6 Preregistered Downstream Test (E1)

E1 asks whether channel differences change agent outcomes.

Design (frozen before the first episode). We used the 14 end-state tasks from §4 with their goals and checkers unchanged: ten hidden-rule tasks and four controls. There were four observation conditions: *real*, *docs-sim*, *A64-sim* (docs plus the active-64 card), and *SIBYL-sim* (docs plus the source card). The simulator is Haiku 4.5 in all simulated conditions. The two focal tasks, receive-cap and price-jump, received 8 replicates per condition, and the other twelve received 2, for 160 episodes in total. We selected the focal tasks using both the channel results and historical failures, so they are not an independent sample. Endpoints: R, pass/fail under verbatim real replay; S, pass/fail from frozen terminal reads issued to the same simulator (the adapter reproduced 88/88 historical verdicts on real reads); F, whether the agent declared completion. **H1 (primary)** was a crossover. SIBYL-sim should beat A64-sim on cap, and A64-sim should beat SIBYL-sim on price, as the teacher-forced results predicted. We tested it as an intersection-union test: both one-sided Fisher tests had to pass at α=.05. Decision language for each outcome was frozen in advance, and secondary endpoints could not rescue H1. The primary cohort was 8 hidden-rule tasks; the two tasks whose goals disclose hidden-rule outputs were excluded.

Primary result. On cap, SIBYL-sim beat A64-sim (8/8 vs. 0/8, p=7.8×10^{-5}). On price, where teacher-forced fidelity favored probing, the two matched (8/8 each), so the prespecified crossover

was not established. The SIBYL simulator returned a capped price on the first update and the agent adapted: exact match scored the value wrong, but success needs only a signal that the rule fired.

Secondary results (descriptive).

- *Repair.* Over the 8 primary tasks, SIBYL-sim improves on docs-sim by $+25.0$ points with no regressions, entirely on those two tasks (0/8 to 8/8 each); task-level sign-flip $p{=}.25$; we claim no repair beyond them.
- *Regression from a wrong rule.* A64-sim scores -18.8 points against docs-sim, regressing on three booking tasks and 3/8 control episodes. The incorrect card item quoted in §5 produced spurious `sold_out` refusals in 12 of 14 booking episodes (0 of 14 in every other condition).
- *False completion and simulator grading.* The agent declared success while real replay failed in 16/28 docs-sim, 8/28 A64-sim, and 0/28 SIBYL-sim or real primary episodes. Simulator grading passed real-replay failures in 16/24, 8/24, and 0/24.

7 Check on Public MCP Servers

Because the testbed's control comes from building it, we use real servers only to test whether its phenomena occur where we wrote nothing: the official git, SQLite, and time MCP reference servers, run over real stdio with frozen plans (Appendix C). They do: reading the code finds 19 undocumented silent or error-surface behaviors. State-blind judges pass 55–61 of 87 doc-only simulator errors ($J{\leq}0.12$). On 44 ordinary tasks written from the documentation, doc-only observations turn 10 of 132 real passes into failures (10 vs. 1, $p{=}.012$), and in 9 the agent claims success; two doc-written checkers were wrong in the same way as both simulators. Here the benefit requires raw code (exact match 39.6 to 58.3%, $p{<}10^{-6}$, largely wording and layout); 7 of the 10 flips miscompute documented query results, which no specification of undocumented behavior fixes.

8 Limitations and Scope

External validity. Our servers are small, author-built, and synthetic, and we wrote the 38 rules. The public-server check (§7) is small, uses one model family and LLM-written calls and tasks, and its repositories may be in pretraining data. We make no claim about the prevalence of undocumented rules in production APIs, or that source compilation scales to multi-service codebases. E1 covers 14 tasks, one agent, and one simulator, with focal tasks chosen from prior data.

Access model and scope. Source arms assume readable source while execution is unavailable or undesirable; given a sandbox, probing is a real alternative, and ours is one policy at two budgets. Adaptive exploration, manual building (Chen et al., 2024), fuzzing, or specification mining could narrow the gap. We measure evaluation only, not policies trained on simulated rewards, and audit only binary, non-probing judges. Cards gave real-backend agents no benefit (29/32 vs. 25/32).

9 Conclusion

The documentation gap is by construction; its anatomy is not: doc-only simulators miss the undocumented rules that contradict what the documentation suggests, reference-free checks do not see these errors, and the errors change what agents do, here and on public MCP servers. Offline source reading taught the most correct and fewest wrong rules, whether summarized (testbed) or raw (public servers). Downstream, source-derived cards repaired both focal failures, success depended on whether a rule fired rather than on its exact value, and one wrong probing rule broke control tasks. Three practices follow: give simulators readable backend source, audit specifications against it, and grade by real replay or paired references rather than plausibility.

Reproducibility Statement

The supplement contains the three servers, their documentation, the frozen rule inventory and demonstration seeds, all conditioning artifacts (cards, summaries, gold descriptions), all prompts, the raw per-item records, and the scripts that recompute every table from them. The rule inventory, E1 tasks, checkers, adapter, analysis script, and cards were hash-frozen before use. Model identifiers are logged per call. E1 cost $9.44 in API usage. We report only results whose raw per-item records and exact conditioning artifacts we hold and recomputed for this version; earlier exploratory measurements that fail this bar (adaptive probing, source masking, a multi-file variant, weight internalization, one full-set judge run) are omitted. The real-world pilot (Appendix C) includes its frozen plans, call sets, recorded real responses, catalogue, artifacts, per-call rows, judge outputs, and the downstream tasks, checkers, and per-episode rows.

AI Use Statement

In addition to the experimental LLM roles described in the paper, we used LLM-based tools to generate the evaluation calls and tasks for the real-world pilot (Appendix C), assist with code debugging and monitoring experiment execution under human supervision, and check the existence and bibliographic accuracy of cited references. The human authors wrote the initial manuscript; LLM-based tools subsequently assisted with proofreading, checking consistency of terminology and claims across the manuscript, and improving style and grammar. We also used LLM-based tools to obtain feedback on experimental methodology, interpretation of results, and manuscript organization. The human authors directed the research and retained final decision-making authority, and take responsibility for the final manuscript, results, code, and data, including all AI-assisted content.

References

Tommaso Castellani, Naimeng Ye, Daksh Mittal, Thomson Yen, Emmanouil Koukoumidis, William Zeng, and Hongseok Namkoong. SynthTools: A framework for scaling synthetic tools for agent development. *arXiv preprint arXiv:2511.09572*, 2025. doi: 10.48550/arXiv.2511.09572.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Proceedings of the Conference on Robot Learning*, pp. 66–75, 2020.

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

Sanjiban Choudhury and Paloma Sodhi. Better than your teacher: LLM agents that learn from privileged AI feedback. *arXiv preprint arXiv:2410.05434*, 2024.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A survey on LLM-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11–16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 11143–11156. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.664. URL https://doi.org/10.18653/v1/2024.findings-acl.664.

Zhicheng Guo, Sijie Cheng, Yuchen Niu, Hao Wang, Sicheng Zhou, Wenbing Huang, and Yang Liu. StableToolBench-MirrorAPI: Modeling tool environments as mirrors of 7,000+ real-world APIs. *arXiv preprint arXiv:2503.20527*, 2025.

Lakshya Gupta, Litao Li, Yizhe Liu, Sriram Ganapathi Subramanian, Kaheer Suleman, Zichen Zhang, Haoye Lu, and Sumit Pasupalak. World of workflows: A benchmark for bringing world models to enterprise systems. arXiv:2601.22130, 2026.

Bhaskar Gurram. Auditing automated evaluation, error propagation, and runtime mitigation in tool-using language agents. *arXiv preprint arXiv:2604.16706*, 2026.

Mohammed Mehedi Hasan, Hao Li, Gopi Krishnan Rajbahadur, Bram Adams, and Ahmed E. Hassan. Model context protocol (MCP) tool descriptions are smelly! Towards improving AI agent efficiency with augmented MCP tool descriptions. *arXiv preprint arXiv:2602.14878*, 2026.

Youling Huang, Guanqiao Chen, Junchi Yao, Lu Wang, Fangkai Yang, Chao Du, ChenZhuo Zhao, Pu Zhao, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Beyond state consistency: Behavior consistency in text-based world models. *arXiv preprint arXiv:2604.13824*, 2026.

Maria Kechagia and Diomidis Spinellis. Undocumented and unchecked: Exceptions that spell trouble. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*, pp. 312–315, 2014. doi: 10.1145/2597073.2597089.

Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025. doi: 10.48550/arXiv.2507.20534.

Yuetai Li, Huseyin A. Inan, Xiang Yue, Wei-Ning Chen, Lukas Wutschitz, Janardhan Kulkarni, Radha Poovendran, Robert Sim, and Saravan Rajmohan. Simulating environments with reasoning models for agent training. *arXiv preprint arXiv:2511.01824*, 2025. doi: 10.48550/arXiv.2511.01824.

Jiateng Liu, Zhenhailong Wang, Xiaojiang Huang, Yingjie Li, Xing Fan, Xiang Li, Chenlei Guo, Ruhi Sarikaya, and Heng Ji. Analyzing and internalizing complex policy documents for LLM agents. *arXiv preprint arXiv:2510.11588*, 2025a.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. ToolACE: Winning the points of LLM function calling. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025b. URL https://openreview.net/forum?id=8EB8k6DdCU.

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, ICLR 2024, Vienna, Austria, May 7–11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Yang Liu, Shiwei Hou, Xiyuan Chen, Yu Wang, Sen Yuan, Qirui Gan, Shao You, Feifan Chen, Wencheng Li, Shuyang Hu, Yongzhou Liu, Emma Xia, Xiaojing Lu, Hao Wang, Fan Xu, and Yanfeng Li. ZhuLong: Execution-grounded LLM agent for EDA scripting with offline API self-exploration. *arXiv preprint arXiv:2608.07925*, 2026a.

Yi Liu, TingFeng Hui, Wei Zhang, Li Sun, Ningxin Su, Jian Wang, and Sen Su. Envsimbench: A benchmark for evaluating and improving llm-based environment simulation. *arXiv preprint arXiv:2605.07247*, 2026b.

Lezhi Ma, Shangqing Liu, Yi Li, Xiaofei Xie, and Lei Bu. SpecGen: Automated generation of formal program specifications via large language models. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, 2025. doi: 10.1109/ICSE55347.2025.00129.

Murong Ma, Tianyu Chen, Yun Lin, Shuai Lu, Qinglin Zhu, Yeyun Gong, Zhiyong Huang, Peng Cheng, Yan Lu, and Jin Song Dong. From patches to trajectories: Privileged process supervision for software-engineering agents. arXiv:2605.21996, 2026.

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

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

Gregory Polyakov, Ilseyar Alimova, Dmitry Abulkhanov, Ivan Sedykh, Andrey Bout, Sergey Nikolenko, and Irina Piontkovskaya. ToolReflection: Improving large language models for real-world API calls with self-generated data. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pp. 184–199. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.realm-1.14.

Zhenzhen Ren, Xinpeng Zhang, Zhenxing Qian, Yan Gao, Yu Shi, Shuxin Zheng, and Jiyan He. GTM: Simulating the world of tools for AI agents. *arXiv preprint arXiv:2512.04535*, 2025. doi: 10.48550/arXiv.2512.04535.

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, ICLR 2024, Vienna, Austria, May 7–11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=GEcwtMk1uA.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023.

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. doi: 10.48550/arXiv.2306.05301.

Gias Uddin and Martin P. Robillard. How API documentation fails. *IEEE Software*, 32(4):68–75, 2015. doi: 10.1109/MS.2014.80.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11–16, 2024*, pp. 9440–9450. Association for Computational Linguistics, 2024a. doi: 10.18653/V1/2024.ACL-LONG.511. URL https://doi.org/10.18653/v1/2024.acl-long.511.

Renxi Wang, Haonan Li, Xudong Han, Yixuan Zhang, and Timothy Baldwin. Learning from failure: Integrating negative examples when fine-tuning large language models as agents. arXiv:2402.11651, 2024b.

Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Shah, Yujia Bao, and Eugene Siow. MCP-Bench: Benchmarking tool-using LLM agents with complex real-world tasks via MCP servers. *arXiv preprint arXiv:2508.20453*, 2025.

Shihao Xia, Mengting He, Haomin Jia, and Linhai Song. Doc2Spec: Synthesizing formal programming specifications from natural language via grammar induction. *arXiv preprint arXiv:2602.04892*, 2026.

Wenkai Yang, Yankai Lin, Jie Zhou, and Ji-Rong Wen. Distilling rule-based knowledge into large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 913–932, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

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.

Bowen Ye, Lei Li, Shicheng Li, Zihao Yue, Linghao Zhang, Hanglong Lv, Yuanxin Liu, Wenhan Ma, Hao Tian, Rang Li, Jinhao Dong, Yikai Zhao, Xiangwei Deng, Hailin Zhang, Liang Zhao, Qi Liu, Lingpeng Kong, Tong Yang, and Fuli Luo. CodeMidas: Scaling agentic coding RL environments from code itself. *arXiv preprint arXiv:2609.22068*, 2026.

XinRan Zhang. MIRAGE: Online LLM simulation for microservice dependency testing. *arXiv preprint arXiv:2604.04806*, 2026.

Zhen Zhang, Kaiqiang Song, Xun Wang, Yebowen Hu, Weixiang Yan, Chenyang Zhao, Henry Peng Zou, Haoyun Deng, Sathish Reddy Indurthi, Shujian Liu, Simin Ma, Xiaoyang Wang, Xin Eric Wang, and Song Wang. CM2: Reinforcement learning with checklist rewards for multi-turn and multi-step agentic tool use. *arXiv preprint arXiv:2602.12268*, 2026. doi: 10.48550/arXiv.2602.12268.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 – 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Hao Zhong and Zhendong Su. Detecting API documentation errors. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications (OOPSLA)*, pp. 803–816, 2013. doi: 10.1145/2509136.2509523.

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. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7–11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Xuhui Zhou, Hyunwoo Kim, Faeze Brahman, Liwei Jiang, Hao Zhu, Ximing Lu, Frank F. Xu, Bill Yuchen Lin, Yejin Choi, Niloofar Mireshghallah, Ronan Le Bras, and Maarten Sap. HAICOSYSTEM: An ecosystem for sandboxing safety risks in interactive AI agents. In *Conference on Language Modeling (COLM)*, 2025.

Yu Zhou, Ruihang Gu, Taolue Chen, Zhiqiu Huang, Sebastiano Panichella, and Harald Gall. Analyzing APIs documentation and code to detect directive defects. In *Proceedings of the 39th IEEE/ACM International Conference on Software Engineering (ICSE)*, pp. 27–37, 2017. doi: 10.1109/ICSE.2017.11.

Yuxin Zuo, Zikai Xiao, Li Sheng, Fei Huang, Jianhong Tu, Yuxuan Liu, Tianyi Tang, Xiaomeng Hu, Yang Su, Qingfeng Lan, Yantao Liu, Qin Zhu, Yinger Zhang, Bowen Yu, Haiquan Zhao, Haiyang Xu, Jianxin Yang, Jiayang Cheng, Junyang Wang, Lianghao Deng, Mingfeng Xue, Tianyi Bai, Yang Fan, Yubo Ma, Yucheng Li, Zeyu Cui, Zhihai Wang, Zhihui Xie, Zhuorui Ye, An Yang, Dayiheng Liu, Jingren Zhou, and Ning Ding. Qwen-agentworld: Language world models for general agents. *CoRR*, abs/2606.24597, 2026.

A Per-Rule Results

B Robustness Replications

These replications were planned and frozen before any call was made, and every arm is reported. The endpoint, evaluation plan, student prompt, and statistics are those of §5.

Rule	Class	n	Docs	K32	A64	A128	SIBYL	Summ.	Raw	Gold
bk_add_guest_dup	error	2	0	0	0	2	2	2	2	2
bk_add_room_dup	error	2	0	0	0	0	2	2	2	2
bk_book_gold_exempt	silent	2	0	0	0	0	2	1	1	1
bk_book_golddiscount	silent	2	0	0	0	0	1	2	1	1
bk_book_maxnights	silent	2	0	0	0	0	0	0	0	1
bk_book_singlenight	silent	2	0	0	2	2	2	1	1	2
bk_book_soldout	error	2	0	0	2	2	2	2	2	2
bk_book_weeklong	silent	2	0	0	0	0	1	1	1	0
bk_book_zeronights	error	2	0	0	2	0	2	2	2	2
bk_cancel_checkedin	error	2	0	0	2	1	2	2	2	2
bk_cancel_out	error	2	0	0	2	0	2	2	2	2
bk_checkin_canceled	error	2	2	2	2	2	2	2	0	2
bk_checkin_double	silent	2	0	0	2	0	2	2	2	2
bk_checkout_notin	error	2	0	0	2	0	2	2	2	2
bk_report_revenue†	derived	4	2	4	1	3	4	4	4	4
inv_add_dup_name	error	2	0	0	0	0	2	2	2	2
inv_cancel_double	silent	2	0	0	2	2	2	2	2	2
inv_cancel_unshipped_noop	silent	2	2	2	2	2	2	2	2	2
inv_discontinue_stocked	error	2	0	0	0	0	0	0	2	0
inv_discontinue_twice	silent	2	0	0	2	2	2	2	2	2
inv_order_discontinued	error	2	2	2	2	2	2	2	2	2
inv_order_partial	silent	2	2	0	0	2	1	2	2	2
inv_order_zero_stock	silent	2	0	0	0	0	0	2	0	2
inv_price_floor	silent	2	0	0	1	2	2	0	2	2
inv_price_jump	silent	2	0	0	2	2	0	2	0	1
inv_receive_cap	silent	2	0	0	0	0	2	0	1	2
inv_receive_discontinued	error	2	1	1	2	2	2	2	2	2
inv_receive_nonpositive	silent	2	0	0	2	2	2	2	2	2
inv_report_reorder_flag†	derived	4	2	4	3	4	4	4	4	4
inv_setprice_discontinued	error	2	2	2	2	2	0	1	2	2
ph_hours_limit	error	2	0	0	2	0	2	2	2	2

Table 5: Per-rule first-encounter correct counts (out of n) for the 31 never-triggered rules. †excluded from the strict-29 cohort. Summ. = generic source-to-documentation summary; Raw = raw-source prompting; Gold = frozen rule descriptions.

Server-level uncertainty. A two-stage bootstrap that resamples the three servers and then rules within each gives 95% intervals of [−5.9, 41.1] for SIBYL − active-64, [0.0, 39.5] for generic summary − active-64, and [25.8, 100] for SIBYL − docs. With three servers, the source-over-probing contrast is not established beyond this inventory.

C Real-World Pilot on Public MCP Servers

To check whether the documentation gap exists outside our testbed, we ran a small pilot on three official Model Context Protocol reference servers that we did not write: `mcp-server-git` 2026.8.18, `mcp-server-sqlite` 2025.4.25, and `mcp-server-time` 2026.8.18, pinned from PyPI. Each ran as a real stdio MCP subprocess on a fixed seeded state, with the clock and git identity frozen. The documentation given to every arm is the README tool section plus the server's own `tools/list` output. Sonnet 5, given only this documentation, wrote 8 episodes of 6 calls per server, about half aimed at edge cases. This gives 144 calls, 38 of which return a real error; every call replays identically from a fresh state. The call set, the catalogue of undocumented behaviors, the arms, the endpoints, and the statistics were frozen before any simulator call. The student is Haiku 4.5 with the settings of §3, teacher-forced on the real episode prefix, one sample per call. It must output the response as `{isError, text}`. Arms add, to the documentation, the generic summary of

Table 6: Cross-family student. The same frozen artifacts as Table 2, with DeepSeek-V3 as the student in place of Haiku 4.5 (0 parse failures). NT-31: correct/observations (%).

Arm	NT-31	rule mean	non-rule
docs only	14/66 (21.2)	20.2	86.8
docs-notes	12/66 (18.2)	15.3	73.7
passive logs K=32	17/66 (25.8)	22.6	86.8
trajectory-PI	16/66 (24.2)	20.2	89.5
active probing, 64	43/66 (65.2)	66.9	90.4
active probing, 128	30/66 (45.5)	44.4	86.0
rule enumeration (SIBYL)	52/66 (78.8)	80.6	85.1
same, Kimi teacher	52/66 (78.8)	79.8	86.0
generic summary	49/66 (74.2)	75.8	93.0
raw source in prompt	49/66 (74.2)	73.4	90.4
gold rule descriptions	55/66 (83.3)	83.9	86.8

SIBYL − active-64: +13.7, p=.076, CI [−3.2, 30.6]
generic summary − active-64: +8.9, p=.15, CI [−4.8, 23.4]
raw source − active-64: +6.5, p=.27, CI [−11.3, 24.2]
SIBYL − docs: +60.5, p<.001; active-64 − docs: +46.8, p<.001

Table 7: Artifact redraws. Two fresh Sonnet 5 draws per arm with unchanged prompts and contamination filter (fresh 64-call probing sessions for active-64), Haiku 4.5 student. Draw 0 is the artifact in Table 2. Entries: NT-31 correct/66, rule mean in parentheses. All source draws passed the filter; 0 parse failures.

Arm	draw 0	draw 1	draw 2	rule mean, 3 draws
rule enumeration (SIBYL)	53 (79.0)	55 (82.3)	53 (79.8)	80.4
generic summary	54 (80.6)	57 (85.5)	44 (64.5)	76.9
active probing, 64	39 (59.7)	30 (43.5)	31 (46.0)	49.7

Draw-averaged per-rule contrasts: SIBYL − active-64 +30.6, p<.001, CI [17.2, 44.1]; generic summary − active-64 +27.2, p<.001, CI [15.3, 39.5].

§5 (Sonnet 5 over the comment-stripped server code), or the raw server code itself. A SIBYL arm (the rule-enumeration prompt) was added after the other results had been seen, because the frozen design had not tested it; we report it regardless. The source contains server code only, so behavior determined by GitPython/git, SQLite, `zoneinfo`, or the MCP SDK is not in it.

The servers do enforce undocumented behavior. We compared real responses with the documentation and checked each discrepancy in the source. Before any student output, this gave a catalogue of 24 undocumented behaviors: 5 formatting differences and 19 silent or error-surface behaviors. Of those 19, 12 are in server code, 6 in libraries, and 1 in the SDK. Examples:

- The SQLite server returns database errors, and its own read-only refusals, in results *not* flagged as errors.
- Its read-only guard checks only the query prefix.
- `git_commit` succeeds with nothing staged.
- `git_create_branch` reports success for an existing branch name.
- `git_log` with a date filter misaligns its own fields when more than one commit matches.
- The time server's README example omits a returned field and shows wrong values.

Sixty-six of the 144 calls touch a non-formatting item.

The channel result transfers only in part. Raw server code raises exact match from 39.6% to 58.3% (29 calls gained, 2 lost; p=4.6×10^{-7}). Only 7 of the 29 gains are on calls that exercise a semantic undocumented behavior (7 gained, 1 lost on those 66 calls, not significant); the other 22 are on calls whose only undocumented aspect is formatting or that touch no catalogued behavior, so most of the raw-source gain is verbatim wording and layout rather than hidden rules. Unlike on our

Table 8: Graded scoring of the Haiku rows behind Table 2 (rule means over 31 NT rules). Outcome type: same error code, or success with the same key set. Field accuracy: share of the real response's top-level fields reproduced exactly. SIBYL − active-64 is +17.7 (p=.035) on outcome type and +18.2 (p=.017) on field accuracy; generic summary − active-64 is +11.3 (p=.13) and +15.0 (p=.023).

Arm	exact	outcome type	field accuracy
docs only	21.0	44.4	35.7
passive logs K=32	21.0	50.0	39.0
active probing, 64	59.7	75.8	70.5
active probing, 128	52.4	64.5	60.4
rule enumeration (SIBYL)	79.0	93.5	88.7
generic summary	80.6	87.1	85.5
raw source in prompt	79.0	95.2	90.1
gold rule descriptions	87.1	90.3	89.5

Table 9: Inputs and artifacts in tokens (cl100k), per server (inventory / booking / projecthub). Source arms make one teacher call over the source (plus regenerations if the contamination filter rejects a draft); active probing makes 64 or 128 live calls before its teacher call. Teacher effort is not normalized across channels.

Item	tokens
comment-stripped source	1093 / 1130 / 948
documentation	375 / 333 / 266
passive transcripts, K=32	61761 / 63725 / 60150
docs-notes card	1268 / 1196 / 902
passive-log card (K=32)	929 / 863 / 1339
trajectory-PI card	621 / 1206 / 1170
active-64 card	1670 / 1783 / 1436
active-128 card	1643 / 1257 / 2049
SIBYL card	1461 / 2502 / 1175
generic summary	1475 / 1862 / 1566
gold descriptions	295 / 299 / 114

testbed, the generic summary captures little of this gain (+4.2 points, 10 gained and 4 lost, p=.18). SIBYL is no better (+6.9 exact, p=.14) and lowers outcome agreement (−11.8, p=.002). Most of its losses (21 of 23) come from the student copying the response shape of the card's example exchanges, which the teacher wrote in the raw MCP content-array or bare-payload form. All three SIBYL cards and the time summary were flagged by the contamination filter after four regenerations. The only remaining hits were 5-grams also visible in the documentation ("(e.g.") or the servers' own error strings; we used the last drafts as the frozen rule prescribes.

Error-versus-success outcomes are mostly right from documentation alone (91.7%). In 10 of the docs-only arm's 12 outcome errors, the simulator *invents* a refusal where the real server silently succeeds, the opposite of the dominant direction on our testbed. Most remaining exact-match errors are in verbatim wording and layout, which no document states. Some behaviors are missed by every arm, including empty commits and library-level messages.

Reference-free judges are blind here too. We applied the two judge prompts of §4 unchanged to the 144 docs-only responses (87 inexact, 57 exact controls), with Haiku 4.5 and Sonnet 5, one sample each, under a plan frozen before any judge call. State-blind judges call 55 (Haiku) and 61 (Sonnet) of the 87 errors plausible, with J=0.09 and 0.12 (bootstrap 95% intervals include 0). With the real history, the Haiku judge flags 60% of correct controls (J=0.00). Only the history-aware Sonnet judge separates the classes at all (J=0.28, $[0.17, 0.40]$, 5.3% false positives), catching 9 of the 12 outcome errors but passing 28 of the 41 errors on calls with a semantic undocumented behavior. As on our testbed, documentation-grounded judging does not certify doc-only simulation on real servers; paired real responses do.

Table 10: Real-world pilot: exact match and outcome (error vs. success) agreement, %. Paired contrasts use an exact McNemar test.

	all calls (n=144)		real error (n=38)		undocumented (n=66)	
arm	exact	outcome	exact	outcome	exact	outcome
docs only	39.6	91.7	31.6	94.7	37.9	83.3
+ generic summary	43.8	93.1	31.6	97.4	39.4	86.4
+ SIBYL (post hoc)	46.5	79.9	44.7	92.1	40.9	75.8
+ raw server code	58.3	93.1	44.7	97.4	47.0	84.8

Doc-only simulation changes agent outcomes on real servers. Following §4, Sonnet 5 wrote 44 ordinary tasks (16 git, 16 SQLite, 12 time) from the documentation alone, each with a deterministic checker validated on a reference call sequence; tasks and checkers were hash-frozen before any episode, and a second batch of 22 was declared before it was generated. A Haiku 4.5 agent (at most 10 calls) ran each task with 3 seeds under three observation sources: the live server, the docs-only simulator, and the docs+source simulator. Every episode's calls were replayed on a fresh real server and scored by the same checker, so only the observation source varies. Two tasks fail on the real server in every seed because their doc-written checkers disagree with the server's actual `git_diff` and SQL semantics (both simulators agree with the wrong checker, so a checker validated only in simulation would have been accepted); excluding them, the real backend passes 126/126 episodes, the docs-only simulator 116, and docs+source 113. All paired changes point the same way: 10 vs. 0 episodes for docs-only (exact McNemar p=.002; with the two tasks, 10 vs. 1, p=.012) and 13 vs. 0 for docs+source (p=2.4×10^{-4}), and in 9 of the 10 docs-only failures the agent reports success. The effect sits in 5 of the 44 tasks (task-level sign test p=.06), none on the time server, and the second batch alone (2 vs. 0) does not replicate it at conventional significance. A per-step classifier, added after seeing results, attributes the 10 docs-only flips mechanically. Seven are value errors on documented SQL reads: the simulator miscomputes an aggregate over seed data it was shown (a top spender or a `SUM`), and the agent updates the wrong row or reports a wrong total. Two are an invented `git_commit` refusal followed by a retry that, on the real server, silently makes an empty commit (catalogue item G6). One is `git_diff` with a target, which diffs the working tree rather than the two branches the README suggests. So on real servers doc-only simulation inflates reported success, but mostly through miscomputed documented results rather than hidden rules, and raw source does not reduce the flips.

Caveats. These are popular public repositories, so both models may have seen their source and README during pretraining; the docs-only arm is therefore not blind to the implementation. The catalogue and its tags are our own judgment, fixed before any student output. We ran one student, one sample per call, and 24 LLM-written episodes, and the call writer and student come from the same model family. Some behavior is platform-dependent (case-insensitive time-zone names on macOS). We treat the pilot as descriptive. It shows that real servers carry undocumented behavior that documentation-conditioned simulation misses. It does not show that a source-derived specification repairs it at our testbed's rates.

D PROMPTS

Generic summary prompt (verbatim task text). "Write complete developer documentation for this server's tools, for a colleague who will use the server but cannot read its code. For each tool, describe its behavior: inputs, outputs, and what happens in the different situations the tool handles. You MUST NOT quote, paraphrase, or reference code, SQL, variable names, or implementation details. Plain text." It is preceded by the documentation and the comment-stripped source.

Card injection in E1. Cards are appended to the simulator prompt under the header "Server behavior notes (from an internal analysis; treat as reliable):". The same header is used for both card conditions.

E E1 Freeze and Decision Table

The permitted conclusions were frozen before any episode, as follows.

- Both crossover tests pass: "The prespecified crossover was supported."
- One or neither passes: "The prespecified crossover was not established."
- S improves but R does not: "Terminal-read grading improved; behavioral repair not demonstrated."
- Any regression versus docs-sim: reported in the main text.

"Repair" wording was permitted only on tasks where the winning condition also beat docs-sim. The single stopping rule was a $40 projected-cost cap; it was not reached.