STEER: SCALING LONG-TERM MEMORY FOR SELF-EVOLVING GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI agents accumulate experience through repeated interaction, but turning that experience into useful long-term memory requires more than retaining past trajectories. Successful procedures and failure-derived lessons need explicit applicability conditions, evidence for routine reuse, and selective activation in later tasks. We present **S**calable **T**ask-conditioned **E**xperience **E**volution and **R**euse (**STEER**), a framework that evolves operational knowledge while keeping the execution agent frozen. STEER represents experience as outcome-conditioned behavioral hypotheses, giving generation, validation, and retrieval a shared operational interface. Consolidation and controlled trials govern admission to a compact deployed store, while TaskContract-based retrieval filters conflicting knowledge and composes complementary procedures and warnings. Both candidate testing and deployed reuse contribute to task execution and further learning. On MobileWorld, STEER raises Seed-2.0-Pro's five-pass cumulative success from 67.5% without memory to 77.8%. Independent evolution on OSWorld V2 also improves task progress. These results demonstrate the value of a complete knowledge lifecycle for improving GUI agents through continued interaction. An anonymized project page is available at https://anonymous.4open.science/w/STEER-GUI/.

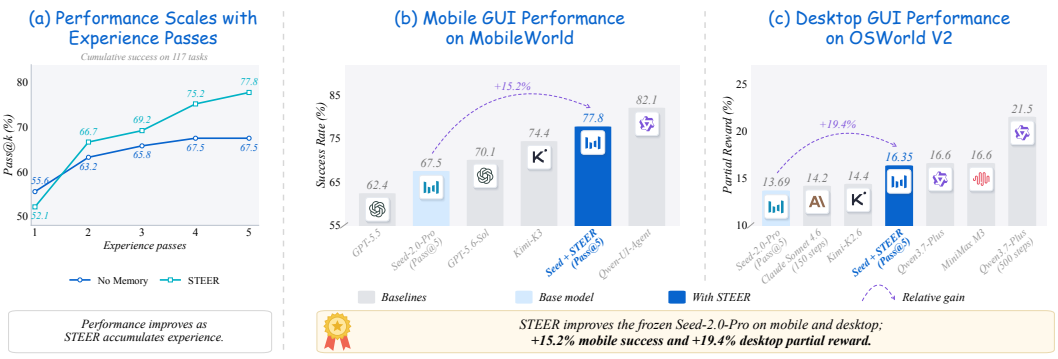

Figure 1: **STEER improves mobile and desktop performance.** The curve shows cumulative success across five passes. Arrows show relative gains over our base agent after five passes; vertical axes are truncated. Public models provide context under their reported settings, not a matched-budget ranking (MobileWorld Team, 2026; XLANG Lab, 2026).

1 INTRODUCTION

GUI agents repeatedly encounter related operations across applications (Rawles et al., 2024; Kong et al., 2026). Successful trajectories contain reusable procedures; failures expose prerequisites and operational pitfalls. External memory retains this experience without retraining the executor. The challenge is to turn continued interaction into better task performance while controlling how much knowledge enters routine reuse and task context.

Prior work provides important building blocks: Reflexion records verbal feedback, ExpeL extracts experience-based insights, and Agent Workflow Memory induces reusable routines (Shinn et al.,

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.

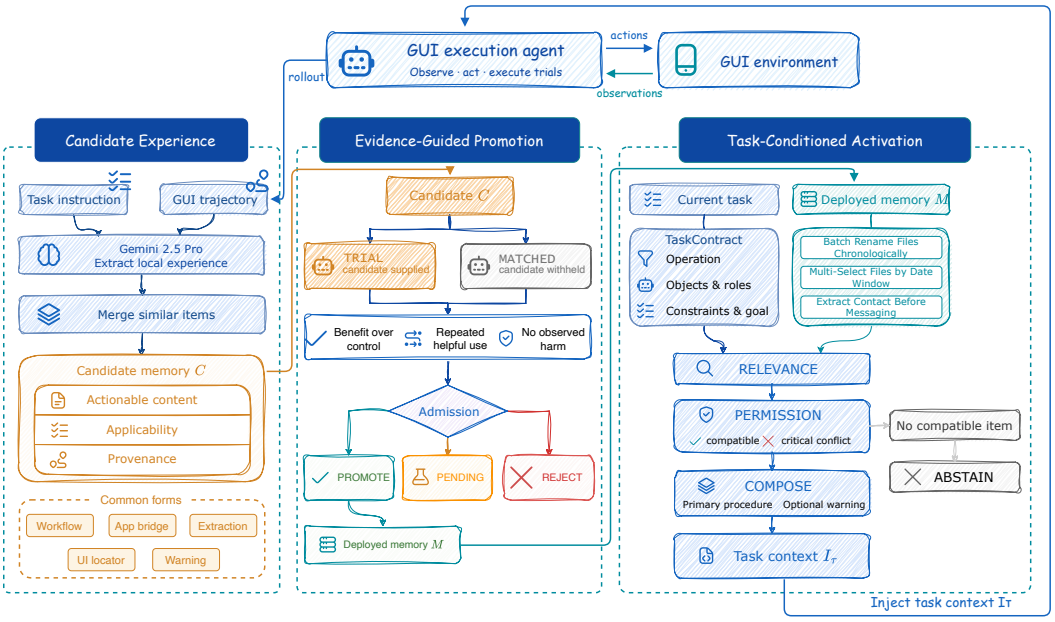

Figure 2: **STEER's execution–experience loop.** Agent trajectories become typed candidate knowledge; controlled trials govern admission to deployed memory, and TaskContract compatibility governs injection. The agent generates experience, tests candidates, and reuses compatible knowledge throughout the loop.

2023; Zhao et al., 2024; Wang et al., 2025b). ReMe distills success patterns and failure triggers, while SkillOpt evaluates local skill edits (Cao et al., 2026; Yang et al., 2026). Generating a plausible instruction does not establish its reusable scope, and recalling a related instruction does not establish its applicability to the current operation.

Three challenges connect experience generation to reuse. First, trajectory outcomes must become operational hypotheses with explicit conditions. Second, hypotheses need testing before admission to routine reuse. Third, retrieval must check action-level compatibility: a recipient-extraction procedure can be relevant to email yet unnecessary when the recipient is already specified.

> ***Research question.*** *How can a GUI agent turn continued interaction into reusable knowledge that improves task performance, while controlling admission to memory and the experience supplied to each task?*

To address this question, we introduce **STEER** (Figure 2), a persistent execution–experience loop. Outcome-conditioned knowledge specifies behavioral roles, applicability, completion conditions, and provenance shared by validation and retrieval. Local consolidation and controlled trials govern admission to deployed memory. TaskContract-based selection checks operational conflicts and composes a primary procedure with a complementary warning, or abstains. Both candidate testing and deployed reuse generate further experience, while the execution policy remains frozen.

On MobileWorld, STEER improves Seed-2.0-Pro's five-pass cumulative success from 67.5% without memory to 77.8% (Figure 1). Three independent comparisons against append-all similarity memory further show consistently higher coverage with 70.5% fewer retrieved-context tokens on average (Table 1). Independently initialized memory also improves task progress on OSWorld V2. Together, these results show that continued experience can improve agent performance through controlled deployment and selective use.

Contributions. (1) We propose an outcome-conditioned operational knowledge representation that connects trajectory evidence to reusable behavioral hypotheses with explicit applicability and provenance. (2) We develop a controlled evolution lifecycle that tests new experience while regulating admission to the routinely reused knowledge store. (3) We propose TaskContract-based retrieval that resolves the mismatch between topical relevance and operational applicability through

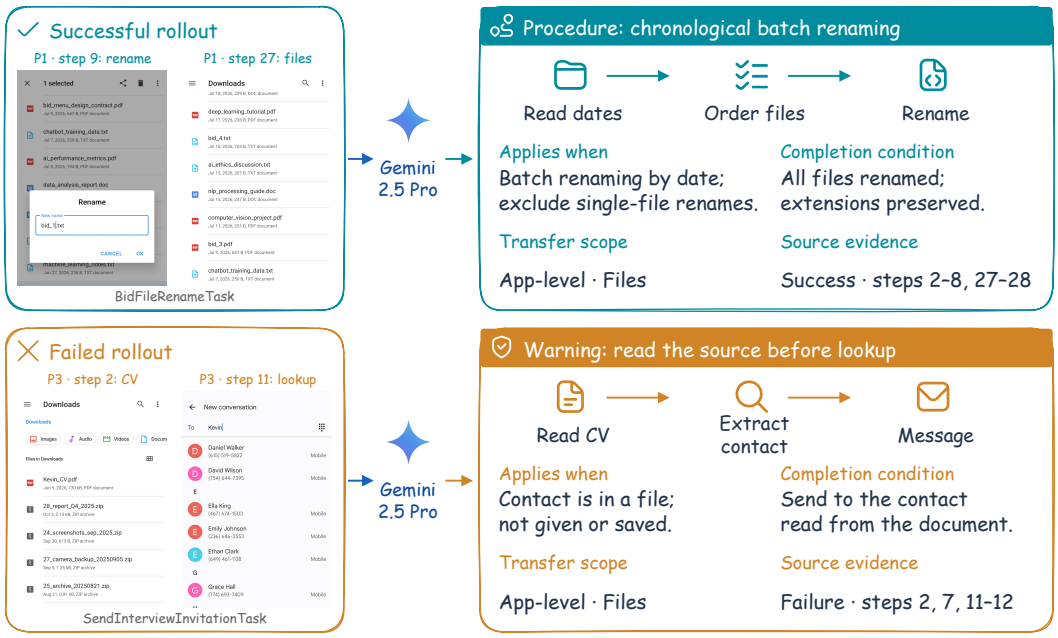

Figure 3: **Operational knowledge from real GUI cases.** Rollout screenshots ground a renaming procedure and a contact-lookup warning. Both retain behavior, applicability, completion conditions, scope, and provenance. Knowledge fields are condensed from recorded items.

conflict filtering and complementary knowledge composition. (4) We show that STEER improves cumulative task coverage and context efficiency on MobileWorld, benefits selection under matched exposure, and supports independent memory evolution on OSWorld V2.

2 Problem Setup

A task τ consists of an instruction x_τ, an initialized GUI environment, and a scorer. A frozen agent π executes a trajectory $\xi_\tau = (o_1, a_1, \ldots, o_T, a_T)$ and receives $y_\tau \in [0, 1]$. Tasks are executed over passes $p = 1, 2, \ldots$; knowledge persists across passes while task environments are reinitialized. We evaluate the entire lifecycle, including executions with candidate trials, deployed memory, or no injected experience.

Let C_p be candidate experience and M_p deployed memory at the start of pass p. The task-specific context I_τ contains selected knowledge: a compatible candidate under a controlled trial, or compatible deployed items during ordinary reuse. A control withholds the candidate being tested. Supplying a candidate does not itself change its deployment state.

We seek increasing task utility while controlling the reusable knowledge consumed by the agent:

$$\max \ \mathbb{E}_\tau[U(\pi, \tau, I_\tau)] - \mu \,\mathrm{tokens}(I_\tau) - \nu \,|M_p|.$$

Here μ, ν express the trade-off rather than learned optimization parameters. Candidate collection can continue outside M_p. We consequently measure task coverage, deployed-memory size, and injected context separately, and inspect candidate growth when assessing the longer lifecycle.

3 STEER

3.1 Outcome-Conditioned Operational Knowledge

STEER converts completed trajectories into typed operational hypotheses. Figure 3 illustrates a successful renaming procedure and a failure-derived warning. Each item specifies actionable content, applicability, intended completion, transfer scope, and provenance, making its behavior and usage boundary explicit.

Formally, an item is $m = (\mathrm{id}, t, s, z, b, p)$: t denotes its functional type, s its transfer scope, z operational contract tags, b conditional behavioral content, and p provenance. Preconditions, exclusions, and postconditions are retained with the behavior. This separates the meaning of a memory from its lifecycle status: a warning remains a warning whether it is pending or deployed, while admission determines when it can be used routinely.

Type determines the behavioral role used for composition; scope constrains testing and retrieval. File selection and recipient extraction can share an application scope while serving different operations, whereas an application bridge can span apps while serving one operation.

From outcomes to knowledge roles. Successful trajectories may propose procedures, positive patterns, bridges, locators, extraction rules, or sink actions. Failed trajectories default to risk rules and negative cases: an observed failure supports a warning about a behavior more directly than it supports a successful replacement procedure. Source outcome therefore determines the default generation permission, not automatic acceptance. Failure-derived recovery strategies can still be represented as hypotheses when that permission is relaxed and subsequently tested.

A shared operational interface. Type identifies compatible merge partners; applicability limits candidate trials; provenance supports evidence tracing; completion conditions specify the intended behavior. In the real `CVEmailTask` example (Figure 4), date-window selection knowledge retains the requested file criterion and attachment operation, allowing the same item to be revised, validated, and later supplied to the executor.

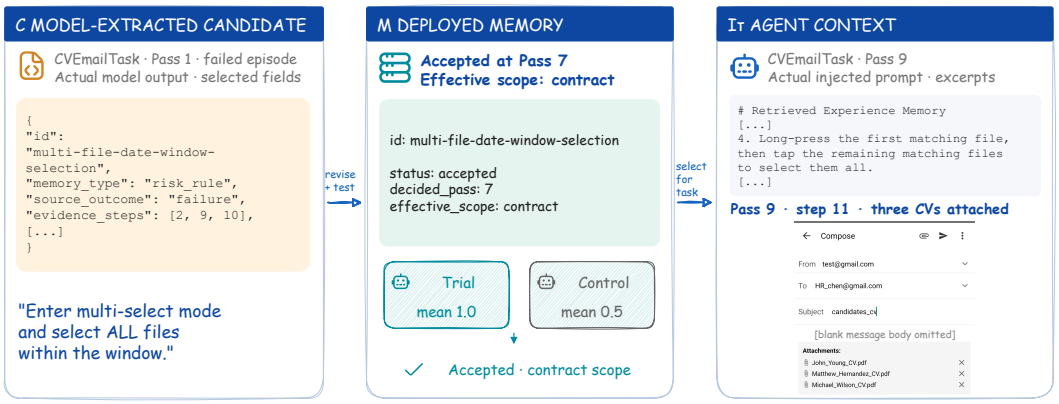

Figure 4: **One knowledge item across its lifecycle.** A recorded Pass-1 output proposes date-window file selection. After revision and controlled testing, the item is accepted at Pass 7 and supplied to `CVEmailTask` in Pass 9. The email GUI shows the recipient, subject, and three CVs; blank body space is omitted. Output and prompt excerpts preserve their recorded wording; omissions are marked.

3.2 Controlled Experience Evolution

Local updates and consolidation. The generator proposes *add*, *replace*, or *delete* edits to individual items. Newly generated or revised knowledge enters C, while an incumbent remains available until a replacement or deletion is accepted. Consolidation merges compatible duplicates and their evidence into a canonical item. Proposed transfer scope is kept narrow unless observations from distinct compatible tasks justify broadening it. Appendix H provides the full edit and scope definitions.

Learning through candidate trials. When a subsequent task is compatible, the agent can execute a trial with candidate e or a matched control without it. Both executions are part of the task-solving lifecycle. Let $\mathcal{E}_t(e)$ and $\mathcal{E}_c(e)$ contain the trial and control outcomes. The observed task-score difference is

$$\widehat{\Delta}(e) = \frac{1}{|\mathcal{E}_t(e)|} \sum_{r \in \mathcal{E}_t(e)} y_r - \frac{1}{|\mathcal{E}_c(e)|} \sum_{r \in \mathcal{E}_c(e)} y_r.$$

Table 1: **MobileWorld performance and retrieved-context cost.** Cumulative success is reported as percentages over 117 tasks. The repeated comparison reports mean ± sample SD across three runs; historical R1 provides the four-condition comparison. Context is the total retrieved-context tokens across five passes; deployed-store growth is shown in Figure 6.

Method	Pass@1	Pass@3	Pass@5	Context (k)
Main comparison (R1)				
No Memory	55.6	65.8	67.5	0.0
AppendAll + Similarity	55.6	66.7	70.1	298.1
AppendAll + Operational	56.4	68.4	74.4	178.0
STEER	52.1	69.2	77.8	83.4
Three independent runs				
AppendAll + Similarity	54.1 ± 2.5	66.1 ± 0.5	70.4 ± 0.5	297.5
STEER	54.4 ± 2.6	68.9 ± 1.3	76.4 ± 1.3	87.8

The evidence also records whether the proposed behavior was exercised successfully and whether harm was observed. Thus a score difference is considered together with behavioral evidence, rather than as the generator's own quality estimate.

Admission to deployed memory. The default rule requires at least two trials and two controls, positive $\widehat{\Delta}(e)$, at least two successful uses of the proposed behavior, and no observed harmful case. Candidates without sufficient evidence remain pending; counterevidence can reject them. Only admitted edits change M. For example, the *batch-chronological-file-rename* candidate receives two trials and two controls before contract-scoped admission at Pass 5 (Appendix J).

Tasks in one pass read a common memory snapshot. Consolidation and admission decisions become visible at the next boundary, so later executions consume experience collected during earlier interaction.

3.3 Task-Conditioned Selection and Composition

STEER parses the instruction into an open-vocabulary TaskContract containing applications, source and destination roles, actions, objects, constraints, operational subproblems, and finish gates. Semantic and lexical retrieval first recall potentially relevant items. A second stage checks these items against the contract, removing critical conflicts such as the wrong action direction, recipient, object, or completion requirement.

Let R_τ denote the recalled items and $\mathcal{T}_\tau$ the task contract. Context construction is

$$I_\tau = \mathrm{Compose}\big(\{m \in R_\tau : \mathrm{Compatible}(m, \mathcal{T}_\tau)\}\big).$$

Increasing and decreasing the same setting share vocabulary but conflict in action direction. An extraction rule is unnecessary when the required value is already given. Compatibility uses the item's stated conditions; composition uses its role.

Selection uses the knowledge roles to compose at most one primary procedure and, when useful, one complementary warning. If no item satisfies the operation, STEER abstains. This makes a short context a consequence of task compatibility rather than a fixed requirement to fill a top-k list.

In the Pass-9 `CVEmailTask`, recall returns a date-window selection procedure and a recipient-extraction warning. The first matches the attachment operation; the second is unnecessary because the instruction already gives the recipient. Only the compatible procedure enters I_τ. Appendix J traces this decision from the retrieved items to the actual context.

4 Experiments

4.1 Experimental Setup

Benchmarks and lifecycle protocol. Our main benchmark is the 117-task GUI-only split of MobileWorld (Kong et al., 2026). Seed-2.0-Pro executes the tasks, and Gemini 2.5 Pro extracts knowledge. All memory conditions start from empty stores and run five passes under the same task order

and task-scoring protocol. Candidate trials occupy task executions within these passes; they are included in the reported results. We additionally evaluate independently initialized STEER on the 108 desktop tasks of OSWorld V2 with Seed-2.0-Pro and a different knowledge-model stack (Appendix I.9).

Comparisons. The replicated comparison repeats STEER and *AppendAll + Similarity* three times with independent stochastic executions. Historical R1 additionally includes *No Memory* and *AppendAll + Operational*, which retains generated experience but uses operational selection. We separate these evidence levels in Table 1. Identity-matched selection, source-disjoint execution, and targeted memory-use interventions test complementary aspects of the lifecycle. Complete protocols and paired statistics are in Appendices A and E.

Measures. MobileWorld cumulative Pass@k is the fraction of tasks solved at least once within k passes. For OSWorld, cumulative partial reward averages each task's best score so far; strict completion requires a full score. We measure deployed items and retrieved-context tokens separately from candidate records. Token savings refer to injected experience, not the complete generation and execution cost. Public references in Figure 1 use one attempt, with 50 steps on MobileWorld and 300 steps on OSWorld unless labeled otherwise; the Sonnet reference uses medium reasoning and the other desktop references use thinking. Public model results in Figure 1 provide contextual reference under their published budgets; our controlled comparisons establish STEER's gains.

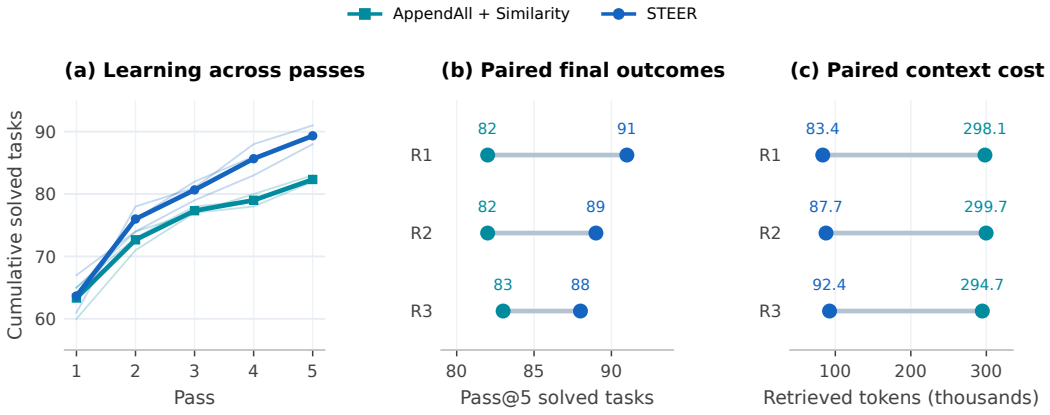

Figure 5: **Consistent gains across three independent MobileWorld runs.** (a) Thin curves show individual runs and thick curves their means. (b) Paired Pass@5 task counts. (c) Retrieved-context tokens. Higher coverage is obtained with less injected experience.

4.2 Improving Performance with Selective Memory

In the main five-pass comparison, STEER solves 91/117 tasks versus 79/117 without memory, raising cumulative success from 67.5% to 77.8%. The four-condition results explain how this advantage develops: append-all similarity memory reaches 82 tasks, operational selection reaches 87, and the complete STEER lifecycle reaches 91. These results motivate separating candidate learning, admission to routine reuse, and task-specific selection.

The replicated comparison confirms higher coverage in all three runs: STEER solves 91, 89, and 88 tasks versus 82, 82, and 83 for append-all similarity memory. Mean cumulative success rises from 70.4% to 76.4%, a gain of seven tasks (5.98 percentage points; task-count sample SD of two). STEER uses 83.4k–92.4k retrieved-context tokens versus 294.7k–299.7k, averaging 70.5% less context. Thus, the improvement over an existing memory baseline is accompanied by substantially less experience entering the agent's prompt.

4.3 Experience Growth and the Deployed Store

STEER separates candidate exploration from admission to the routinely reused store. In historical R1, candidates guide 141 of 585 task episodes during the first five passes. This testing stage is part

of the learning lifecycle and supplies experience before routine deployment. Table 9 and Table 10 report exposure by stage; the main result measures the complete lifecycle, rather than deployed reuse alone.

To inspect the later transition, we extend the descriptive R1 trace to nine complete passes (Figure 6). The deployed set grows from one accepted item after Pass 5 to 12 after Pass 9, while cumulative coverage increases from 91 to 95 tasks. The first nine passes each contain all 117 task results. We use the post-pass admission ledger for memory counts, so the depicted state reflects completed decisions rather than an earlier snapshot.

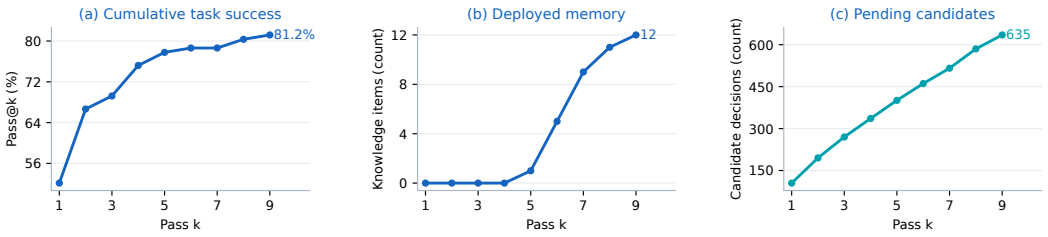

Figure 6: **The lifecycle beyond the initial five passes.** Historical R1 over nine complete passes: cumulative Pass@k over 117 tasks, deployed knowledge items, and pending candidate decisions. Memory counts are measured after each pass's admission decisions. This is a descriptive extension of the five-pass controlled experiment.

Candidate exploration continues outside the deployed store: pending decisions increase from 105 after Pass 1 to 635 after Pass 9. Thus the observed scaling benefit is selective deployment and consumption of accumulated experience. The trace demonstrates compact deployment over the tested horizon, rather than bounded growth of every stored candidate or evidence record.

4.4 Representation and Selection Mechanisms

Outcome-conditioned representation. A controlled MW30 study checks the generation rule. The default representation produces no positive procedures from failed trajectories; removing outcome conditioning produces two, while both conditions solve 19/30 tasks by Pass 3. A targeted probe finds that one failure-derived recipient-extraction hypothesis helps its source task (3/3 versus 0/3), illustrating why such content is usefully treated as a testable hypothesis. These observations support the distinction between source evidence and deployment permission; complete results appear in Appendix E.

Choosing the operationally appropriate item. We freeze the 141 task-pass identities that received candidate trials in historical R1, together with their candidate pools and item counts. Each identity is replayed twice with the STEER-selected item, the top same-pool similarity item, or no memory. This fixes the opportunities for memory use while changing which experience reaches the agent.

Table 2: **Memory selection under controlled exposure.** Top: 282 executions per condition (141 identities, twice). Bottom: 117 tasks over five passes.

Condition	SR (%)	Tokens
Identity-matched replay		
No memory	53.55	0
Same-pool similarity	59.93	131,896
STEER selection	**69.86**	131,290
Source-disjoint cumulative Pass@5		
Similarity	75.21	294,087
STEER	**76.92**	40,100

STEER selection achieves 69.86% success versus 59.93% for similarity and 53.55% without memory (Table 2). Its 9.93-point advantage over similarity has a task-cluster bootstrap 95% CI of $[4.61, 15.60]$ and exact sign-flip $p = 0.0007$. Context costs differ by less than 0.5%. The result supports selecting the appropriate experience rather than merely supplying more text. In a separate frozen retrieval study, operational activation selects 65 items versus 234 for similarity and reduces context characters by 69.7%; Appendix E reports the associated execution tests.

Testing before and after admission. Targeted interventions directly compare supplying versus withholding memory (Figure 8). Across nine predefined candidate-task comparisons, five effects are positive, three neutral, and one negative. Across ten deployed task–memory pairs, four are positive and six neutral. Each arm uses three repetitions. Aggregating these executions, candidate supply raises success from 5/27 (18.5%) to 14/27 (51.9%); deployed reuse raises it from 21/30 (70.0%) to 27/30 (90.0%). The two studies use different selected task sets, not the full benchmark; their full intervals and paired tests are retained in Appendix J.

4.5 Cross-Task Use and Desktop Generalization

Cross-task experience. We prohibit an item from influencing any task present in its provenance and rerun both memory methods. STEER reaches 90/117 tasks versus 88/117 for source-disjoint similarity, using 40.1k versus 294.1k context tokens (Table 2). Its coverage remains within the 88–91 range observed in the standard runs. In the 32 cross-task identities of the matched replay, STEER selection also improves success over no memory by 14.06 points. Together, these observations provide evidence of useful experience beyond exact-source repetition.

Table 3: **Independent memory evolution on OSWorld V2.** Cumulative partial reward and fully solved tasks over five passes.

	Partial reward (%)		Solved / 108	
Pass	Base	STEER	Base	STEER
1	7.90	6.46	0	0
2	9.53	9.07	0	0
3	11.35	12.01	0	0
4	12.88	15.47	0	1
5	13.69	**16.35**	0	**2**

Independent evolution on OSWorld V2. The desktop experiment changes applications, device interaction, and the knowledge-model stack. Memory starts empty in this environment; the shared element is the STEER lifecycle. Table 3 shows that cumulative partial reward overtakes the base agent from Pass 3 and reaches 16.35% at Pass 5, versus 13.69%. Strict completions appear in the later STEER passes, reaching two unique tasks compared with none for the base agent. Partial reward captures progress even when the complete task remains unsolved; the strict-completion columns separately report fully solved tasks. The result supports applying the lifecycle in both mobile and desktop environments, with knowledge learned locally rather than transferred from mobile tasks.

4.6 Knowledge Changes Observable GUI Behavior

The deployed warning in Figure 7 illustrates the operational content behind these measurements. In the historical Pass-8 `SendInterviewInvitationTask`, the agent locates `Kevin_CV.pdf` but searches saved contacts instead of opening the document. In Pass 9, memory directs it to inspect the source resume, extract the phone number, and enter that number directly before sending the SMS. The warning supplies a source-before-contact dependency that is relevant to this task. A separate matched intervention on the same task and memory item, B-09, succeeds in 3/3 reuse repetitions versus 0/3 no-memory repetitions.

5 Related Work

Cross-task experience reuse. External memory converts past interaction into guidance for future tasks without requiring an update to the execution policy. ExpeL extracts insights and Agent Workflow Memory induces reusable routines (Zhao et al., 2024; Wang et al., 2025b); Mobile-Agent-E accumulates tips and reusable shortcuts, while ExpAct organizes structural and procedural experience (Wang et al., 2025a; Tang et al., 2026). Darwinian Memory further regulates reusable units through utility-driven selection and pruning (Mi et al., 2026). STEER focuses on two decisions within that process: when a candidate enters routine reuse and when an available item satisfies the current task's operational conditions.

Long-horizon context management. During a long GUI task, the agent must preserve useful state while limiting redundant history. MobileUse uses hierarchical reflection to detect and recover from execution errors (Li et al., 2025). MemGUI-Agent makes context curation part of the action policy, jointly maintaining folded history, UI state, and recent records (Liu et al., 2026a). MementoGUI

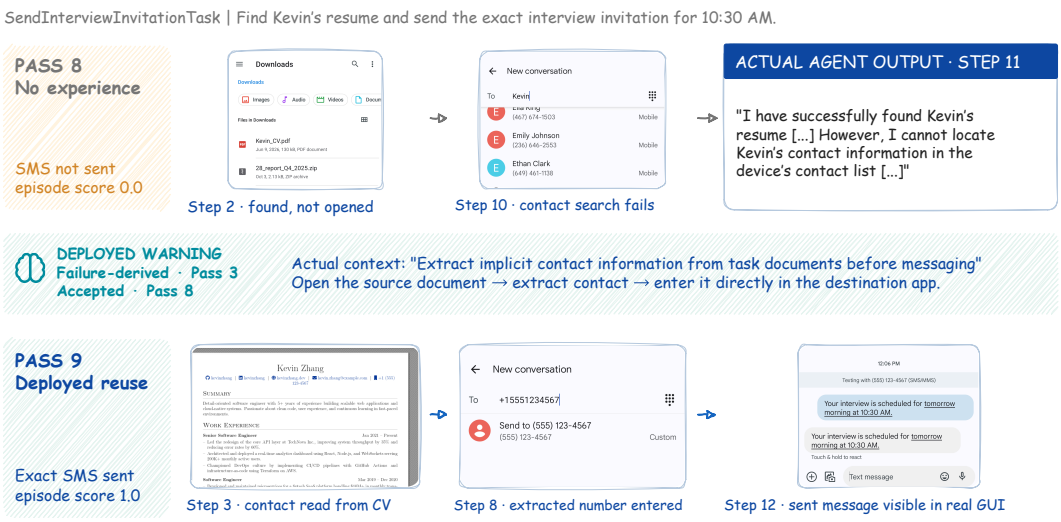

Figure 7: **A deployed warning changes the information-gathering path.** Original Pass-8/9 GUI observations show the difference between searching contacts and inspecting the source resume before sending the SMS. The memory and output excerpts are recorded artifacts. The separate B-09 matched study obtains 3/3 versus 0/3 successes; its repetitions are not the episodes pictured.

combines multimodal working-memory compression with learned episodic selection (Zeng et al., 2026). STEER instead organizes trajectory-derived operational hypotheses across episodes, with explicit testing and admission before routine reuse; it can complement an executor's working-context management.

Interaction-driven policy learning. Another route converts interaction experience into training signals so that improvements are absorbed into model parameters. Memory and corrective hints guide exploration and learning. UI-Mem supplies hierarchical experience guidance at different strengths within online RL rollout groups, encouraging the policy to internalize guided behavior (Xiao et al., 2026). MobileForge connects target-app task generation, hierarchical evaluation, and feedback-guided policy optimization (Liu et al., 2026b). STEER keeps the execution policy frozen and learns through an external knowledge lifecycle, making its adaptation mechanism distinct from these policy-training approaches.



Figure 8: **Aggregate success in targeted memory interventions.** Each panel compares supplying versus withholding memory on its own selected task set, with three executions per arm per pair. These are execution success rates, not full-benchmark Pass@k. Per-pair outcomes appear in Appendix J.

6 Conclusion

STEER turns GUI interaction into reusable operational knowledge through controlled evolution and selective retrieval. MobileWorld experiments and independent OSWorld V2 evolution support this lifecycle: task coverage or progress improves while deployment and context consumption remain selective.

AI USE STATEMENT

Generative AI tools were used in two roles. In the STEER research workflow, LLM-based GUI agents generated task trajectories, while LLM-based knowledge models extracted and revised trajectory-grounded operational knowledge, as described in Sections 3 and 4 and the experimental appendices. Generative AI assistants also supported methodology and experiment discussions, research code development, data processing and analysis, figure preparation, and manuscript editing and LaTeX formatting. The authors reviewed the AI-assisted work, including code, experimental results, figures, citations, and the final manuscript, and take full responsibility for this submission.

REFERENCES

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16803–16822. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.829. URL https://aclanthology.org/2026.findings-acl.829/.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, Zhidan Liu, Steven Hoi, and Yue Wang. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive, and mcp-augmented environments. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026. doi: 10.18653/v1/2026.acl-long.278. URL https://aclanthology.org/2026.acl-long.278/.

Ning Li, Xiangmou Qu, Jiamu Zhou, Muning Wen, Kounianhua Du, Xingyu Lou, Qiuying Peng, Jun Wang, and Weinan Zhang. Mobileuse: A hierarchical reflection-driven gui agent for autonomous mobile operation. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3994410d63ec68ce9a66011a34c9a2c4-Abstract-Conference.html.

Guangyi Liu, Gao Wu, Congxiao Liu, Pengxiang Zhao, Liang Liu, Mading Li, Zhang Qi, Mengyan Wang, Liang Guo, and Yong Liu. Memgui-agent: An end-to-end long-horizon mobile gui agent with proactive context management. *arXiv preprint arXiv:2606.19926*, 2026a. URL https://arxiv.org/abs/2606.19926.

Guangyi Liu, Pengxiang Zhao, Gao Wu, Yiwen Yin, Mading Li, Liang Liu, Congxiao Liu, Zhang Qi, Mengyan Wang, Liang Guo, Jiangning Zhang, and Yong Liu. Mobileforge: Annotation-free adaptation for mobile gui agents with hierarchical feedback-guided policy optimization. *arXiv preprint arXiv:2606.19930*, 2026b. URL https://arxiv.org/abs/2606.19930.

Hongze Mi, Yibo Feng, WenJie Lu, Song Cao, Jinyuan Li, Yanming Li, Xuelin Zhang, Haotian Luo, Songyang Peng, He Cui, Tengfei Tian, Jun Fang, Hua Chai, and Naiqiang Tan. Darwinian memory: A training-free self-regulating memory system for gui agent evolution. *arXiv preprint arXiv:2601.22528*, 2026. doi: 10.48550/arXiv.2601.22528. URL https://arxiv.org/abs/2601.22528.

MobileWorld Team. MobileWorld official leaderboard, 2026. URL https://tongyi-mai.github.io/MobileWorld/. Accessed September 6, 2026. GUI-only public results.

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. *arXiv preprint arXiv:2405.14573*, 2024. doi: 10.48550/arXiv.2405.14573. URL https://arxiv.org/abs/2405.14573.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Shunye Tang, Yuxiang Chai, Yaxuan Guo, Rongduo Han, Mingang Lan, Liang Liu, Guangyi Liu, Pengxiang Zhao, Han Xiao, Qinyi Luo, Guozhi Wang, Hao Wang, Xiao Liang, Haining Zhang, and Wei Wang. Expact: Building and evolving structural and procedural experience for mobile gui agents. OpenReview preprint, 2026. URL https://openreview.net/forum?id=qkSahzB9hh.

Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. Mobile-agent-e: Self-evolving mobile assistant for complex tasks. *arXiv preprint arXiv:2501.11733*, 2025a. doi: 10.48550/arXiv.2501.11733. URL https://arxiv.org/abs/2501.11733.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 63897–63911, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

Han Xiao, Guozhi Wang, Hao Wang, Shilong Liu, Yuxiang Chai, Yue Pan, Yufeng Zhou, Xiaoxin Chen, Yafei Wen, and Hongsheng Li. Ui-mem: Self-evolving experience memory for online reinforcement learning in mobile gui agents. *arXiv preprint arXiv:2602.05832*, 2026. doi: 10.48550/arXiv.2602.05832. URL https://arxiv.org/abs/2602.05832.

XLANG Lab. OSWorld 2.0 official leaderboard, 2026. URL https://osworld-v2.xlang.ai/. Accessed September 6, 2026. Full-set results, release v2026.06.24.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026. doi: 10.48550/arXiv.2605.23904. URL https://arxiv.org/abs/2605.23904.

Ziyun Zeng, Hang Hua, Bocheng Zou, Mu Cai, Rogerio Feris, and Jiebo Luo. Mementogui: Learning agentic multimodal memory control for long-horizon gui agents. *arXiv preprint arXiv:2605.18652*, 2026. URL https://arxiv.org/abs/2605.18652.

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*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

A Replicated Longitudinal Details

Table 4 reports the task-level paired statistics for each independent run. The three differences are directionally consistent, but their individual confidence intervals and tests differ; the paper therefore does not claim that every replicate is independently significant.

Table 4: Per-run paired task statistics for STEER minus AppendAll+Similarity at Pass@5.

Run	Δ tasks	Gains	Losses	Same	95% CI (pp)	p
R1	+9	12	3	102	[+1.71,+14.53]	0.03516
R2	+7	12	5	100	[-0.85,+12.82]	0.14346
R3	+5	10	5	102	[-2.56,+11.11]	0.30176

A.1 Detailed Longitudinal Scaling for the Historical R1 Run

Figure 9 separates cumulative task coverage, memory growth, and retrieved-context cost for the four historical R1 conditions.

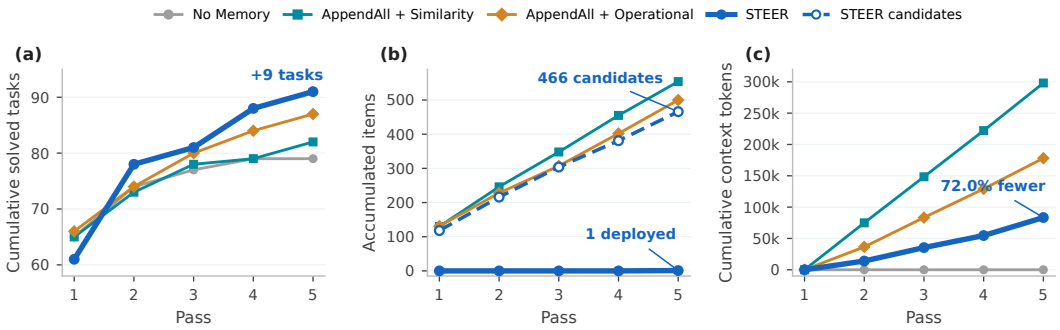

Figure 9: Detailed longitudinal scaling for the historical R1 run. (a) Cumulative task coverage. (b) Experience acquisition and persistent deployed-memory growth. (c) Cumulative experience-context prompt tokens.

A.2 Count-Matched Similarity Control

STEER retrieves memory selectively, so its advantage over append-all similarity memory could simply come from exposing the agent to memory less often. Count-Matched Similarity keeps the append-all store and similarity retriever but uses exactly the same number of memory-injected episodes as STEER in each pass.

Even with the same injection frequency, Count-Matched Similarity solves 78/117 tasks (66.7%) by Pass 5, compared with 91/117 (77.8%) for STEER. The paired difference is 13 tasks (95% CI $[+4.27, +17.95]$ percentage points; McNemar $p = 0.0044$). Both methods use 141 memory-injected episodes in total, so this historical full-run control gives the same qualitative conclusion as the identity-matched replay.

Table 5: MobileWorld exposure control. Count-Matched Similarity and STEER use the same number of memory-injected episodes in every pass.

Method	Injected task-passes	Context tokens	Pass@5
AppendAll + Similarity	468	298,149	82/117
Count-Matched Similarity	141	167,973	78/117
STEER	**141**	**83,362**	**91/117**

Count-Matched Similarity uses more context than STEER (167,973 versus 83,362 tokens), because it retrieves two similarity-ranked items whenever an episode is selected. The token-budget-matched sensitivity analysis below remains supporting rather than primary evidence because its first pass is unusually weak.

B Full-Scale MobileWorld Paired Comparisons

Table 6 reports all three predefined Pass@5 paired comparisons from the frozen Round-2 analysis.

Table 6: Predefined paired Pass@5 comparisons on 117 MobileWorld tasks. Effects are computed as the right method minus the left method.

ID	Comparison	Δ tasks	Δ pp	Gain	Loss	Same	95% CI; p
C1	STEER − AppendAll+Similarity	+9	+7.69	12	3	102	[+1.71,+14.53]; .03516
C2	STEER − AppendAll+Operational	+4	+3.42	9	5	103	[-2.56,+9.40]; .42395
C3	AppendAll+Operational − AppendAll+Similarity	+5	+4.27	9	4	104	[-1.71,+10.26]; .26685

Table 7: Five-pass MobileWorld results. Pass@k is cumulative task coverage; memory and context columns report the state at or through Pass 5.

Method	P@1	P@2	P@3	P@4	P@5	Deployed	Context tokens
No Memory	65	74	77	79	79	0	0
AppendAll + Similarity	65	73	78	79	82	554	298,149
AppendAll + Operational	**66**	74	80	84	87	500	178,042
STEER	61	**78**	**81**	**88**	**91**	**1**	**83,362**

For completeness, the source-disjoint Pass@5 comparison is $+2$ tasks ($+1.71$ percentage points) for SD-STEER relative to SD-Similarity, with nine gains, seven losses, and 101 unchanged tasks (95% CI $[-5.13, +8.55]$; exact McNemar $p = 0.80362$). Within the 32-identity diagnostic cross-task subset, STEER exceeds same-pool similarity by 10.938 percentage points (95% CI $[2.857, 18.750]$; exact sign-flip $p = 0.0625$) and no memory by 14.063 points (95% CI $[4.286, 25.758]$; $p = 0.03125$). These subgroup statistics are supplementary diagnostics rather than a standalone preregistered superiority test.

B.1 Token-Budget-Matched Similarity Sensitivity

Table 8: Token-budget-matched similarity sensitivity on 117 MobileWorld tasks. ExactToken is reported as supporting evidence rather than the primary fairness control.

Method	P@1	P@2	P@3	P@4	P@5	Tokens
ExactToken	47	69	76	79	81	80,340
STEER	61	78	81	88	91	83,362

The nominal final paired comparison gives STEER a 10-task advantage over ExactToken, with 10 gains, zero losses, and 107 unchanged tasks (95% CI [+4.27,+13.68], exact McNemar $p = 0.001953$).

Round 10B finds no deterministic prompt, preflight, or runner bug, but ExactToken's Pass 1 is unusually weak at 47/117. This weak initial realization changes the later append-all store trajectory. The same audit also finds that the historical NoMemory task-order hash, `61eb5d...ef0c`, differs from the hash used by the other main arms, `0c3ad0...46b5`. ExactToken is therefore valid supporting evidence only and is not presented as cleaner than the primary Count-Matched Similarity control.

C Context and Success Provenance

Tables 9 and 10 summarize the frozen Round-4A audit. Across Passes 1–5, all 141 STEER experience-bearing episodes are controlled candidate trials and none contains ordinary deployed-memory reuse. Their 83,362 experience-context prompt tokens exactly reconcile with the execution-service token audit. The first deployed item is promoted only at the Pass-5 boundary, after the reported Pass@5 executions.

Table 9: Per-pass provenance of STEER experience context over five MobileWorld passes.

Pass	Candidate trials	Deployed reuse	No experience	Trial tokens
1	0	0	117	0
2	24	0	93	13,929
3	36	0	81	21,440
4	33	0	84	19,250
5	48	0	69	28,743
Total	141	0	444	83,362

Table 10: **Exposure and first-success provenance in Passes 1–5.** First successes classify each task's earliest successful episode by its context. No deployed-memory exposure occurred in this window, so its effect is not evaluated here. These counts describe exposure, not causal contributions.

Context category	Exposed episodes	First successes
Controlled candidate trial	141	16
Deployed-memory reuse	0	0
Mixed	0	0
No experience	444	75
Total	585	91

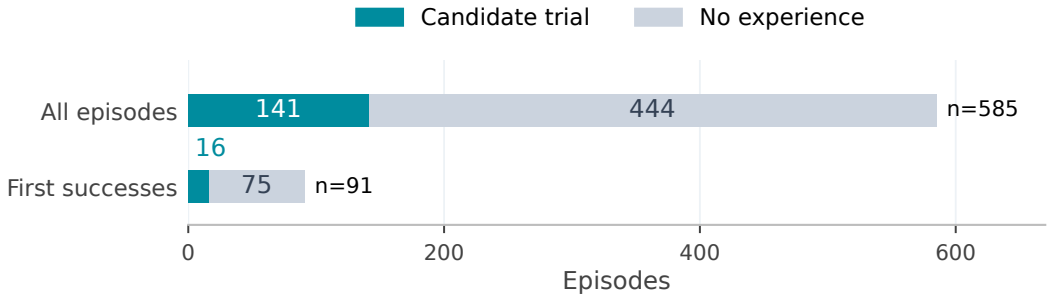

Figure 10: Context provenance for all 585 STEER episodes and the 91 first-success episodes. Controlled candidate trials account for 141 and 16 episodes, respectively; ordinary deployed-memory reuse is zero in both populations.

These provenance counts characterize the context present in first-success episodes and do not constitute causal attribution.

The complete 15-task C1 disagreement table is retained in the machine-readable artifact `steer_c1_disagreement_provenance.csv`; the appendix reports no causal attribution from those context labels.

D Model Sensitivity

The complete MW30 matrix varies both the execution agent and the knowledge generator. With the default generator, memory adds two cumulative tasks for both the default and Kimi execution agents. With the Qwen generator, the corresponding changes are -3 and $+1$. This result supports directional consistency across execution agents under the default generator, but not robustness to an arbitrary knowledge generator.

With the default knowledge generator fixed, paired effects are +0.0667 for both the default and Kimi K2.6 execution agents, with 95% confidence intervals [-0.1000, 0.2333] and [-0.0667, 0.2000], respectively.

Table 11: MW30 model matrix at Pass@3.

Execution agent	Knowledge generator	Memory	Pass@3	vs. matched Base
Default	–	off	17/30	–
Default	Default	full	19/30	+2
Default	Qwen	full	14/30	-3
Kimi K2.6	–	off	6/30	–
Kimi K2.6	Default	full	8/30	+2
Kimi K2.6	Qwen	full	7/30	+1

E COMPLETE MECHANISM TABLES

Tables 12–14 provide the complete controlled results behind the mechanism studies. They cover failure-sourced hypotheses, evidence-guided state transitions, and frozen task-conditioned activation, including null and negative findings.

E.1 A1: OUTCOME-CONDITIONED GENERATION

Table 12 combines the MW30 ablation with targeted validation of the two positive hypotheses sourced from failures.

Table 12: A1 outcome conditioning. The targeted probe forces each failure-sourced positive hypothesis into Canary and omits it from Control.

MW30 arm	Pass@3	Positive from failure
Full	19/30	0
No outcome condition	19/30	2
Targeted hypothesis/task	**Canary**	**Control**
Recipient extraction/source	3/3	0/3
Recipient extraction/transfer	3/3	3/3
Calendar verification/source	0/3	0/3
Calendar verification/transfer	0/3	0/3

Removing outcome conditioning creates two positive hypotheses from failed traces, versus zero under Full, while both arms reach 19/30 cumulative Pass@3. In a targeted 24-episode probe, the recipient-extraction hypothesis changes its source task from 0/3 to 3/3 and triggers the intended recipient extraction and calculation behavior in all three successes. It adds no benefit on the transfer task. The calendar hypothesis changes neither source nor transfer outcomes. The useful conclusion is not that failures cannot yield positive knowledge, but that a failed trace can produce a useful recovery hypothesis whose transfer must still be validated.

E.2 A2: EVIDENCE-GUIDED STATE TRANSITIONS

Table 13: A2 evidence-guided promotion control. The first block holds 20 candidates fixed; the second evaluates 12 promoted and 12 matched rejected items on tasks excluded from their decisions.

Policy	Promoted	Rejected	Trial–control	95% CI
Evidence	0/20	20/20	+.025	[-.125,.175]
Immediate	20/20	0/20	-.025	[-.175,.125]
Ledger group	**Items**	**Mean Δ**	**Pos./neutral/neg.**	**95% CI**
Promoted	12	.000	2/8/2	[-.111,.111]
Rejected	12	-.028	1/8/3	[-.167,.139]

On the identical set of 20 candidates, the evidence policy rejects every item while immediate promotion deploys every item, establishing direct control over deployed-store growth. Trial–control differences are +0.025 and -0.025, respectively. In the additional 144-episode quality probe, the promoted-minus-matched-rejected mean difference is only +0.0278, with a 95% confidence interval of [-0.2222, 0.2500] and exact sign-flip $p = 1.0$. We therefore position the mechanism as conservative promotion and scale control, not as a validated high-value-knowledge classifier.

E.3 A3: FROZEN TASK-CONDITIONED ACTIVATION

Operational $k = 2$ abstains on 68/117 queries, reducing selected items from 234 to 65 and context characters from 607,317 to 184,162, a 69.7% reduction. In the preregistered comparison, similarity $k = 2$ versus operational $k = 2$ has three gains and one loss, effect +0.0833, 95% confidence

Table 14: A3 frozen task-conditioned activation. Offline metrics use 117 queries; GUI columns report cumulative Pass@3 on Retrieval24.

Policy	Activated tasks	Items	Context chars	GUI Pass@3
No activation	–	0	0	20/24
Operational $k = 1$	49/117	61	170,742	20/24
Operational $k = 2$	49/117	65	184,162	19/24
Operational $k = 3$	49/117	65	184,162	22/24
Similarity $k = 2$	117/117	234	607,317	21/24

interval [-0.0833, 0.2500], and McNemar $p = 0.625$. The GUI probe therefore does not distinguish their success rates. Operational $k = 2$ and $k = 3$ have identical offline selections, so their GUI difference is execution noise rather than evidence for choosing k. The directly supported advantage is controlled coverage and context efficiency.

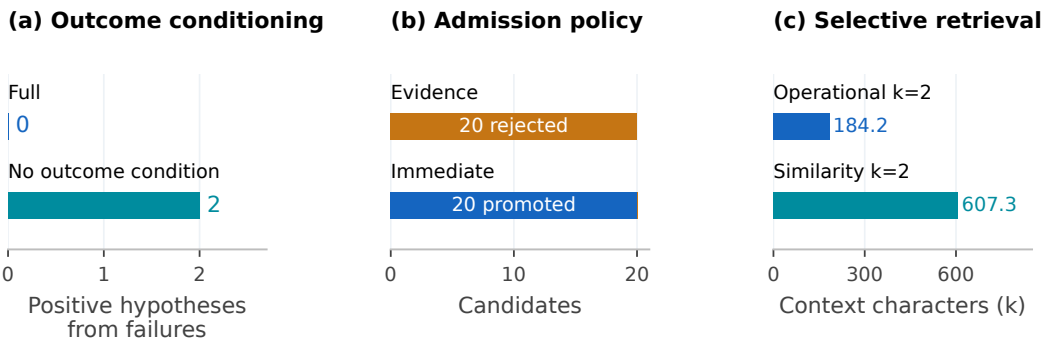

Figure 11: Compact mechanism diagnostics from the complete appendix tables. (a) Removing outcome conditioning produces two positive hypotheses from failed traces, while both conditions remain at 19/30 Pass@3. (b) The evidence and immediate policies change deployment state for the same 20 candidates; this panel measures state control, not knowledge quality. (c) Operational activation reduces context characters relative to similarity retrieval, while the paired GUI comparison is not statistically significant.

F Retrieval Repeat Results

Table 15 reports per-repeat attempt and cumulative results for all five frozen Retrieval24 conditions. All conditions use the same tasks, order, models, image, retry policy, and read-only knowledge pool, with 72 valid episodes per condition.

Table 15: Retrieval24 results across three repeats.

Arm	Attempt R1/R2/R3	Cumulative Pass@1/2/3
Operational $k = 0$	17/16/16	17/19/20
Operational $k = 1$	17/18/19	17/19/20
Operational $k = 2$	17/18/17	17/19/19
Operational $k = 3$	18/19/18	18/21/22
Similarity $k = 2$	18/19/16	18/21/21

Operational $k = 2$ and $k = 3$ have identical offline selected IDs, so their GUI difference cannot be attributed to additional knowledge. An independent Top-1 blind-annotation packet was frozen, but annotation was deferred before labeling began. We create no surrogate labels and report neither applicability precision nor harmful-injection rate.

G TRACEABLE LIFECYCLE CASES

Diagram icons are adapted from Flaticon Uicons (Freepik Company). GUI crops are taken from the recorded experimental trajectories; the diagram layouts, text, arrows, and icons are vector graphics.

Table 16 links each mechanism case to an observable behavior or state decision.

Table 16: Traceable lifecycle cases. Each case links a mechanism to an observable decision or behavior.

Mechanism	Case	Observation
Failure-sourced hypothesis	Recipient extraction from a source document	A controlled candidate trial recovers `CheckInvoiceTask2` in 3/3 trials by extracting the missing address and recomputing the amount; the matched transfer task is already solved in both conditions.
Evidence-guided promotion	Desktop-view extraction for repository statistics	Trial/control evidence is 0/1 and 0/0 on two matched tasks. Evidence-guided promotion rejects it, whereas immediate promotion writes the identical candidate into the deployed store.
Task-conditioned activation	Font-size task versus two flight-mode memories	Similarity-only selects two Settings memories about flight mode. Task-conditioned activation abstains because neither procedure can perform the requested font-size operation.

H DETAILED MEMORY AND RETRIEVAL FORMULATION

H.1 DETAILED MEMORY REPRESENTATION AND EDIT OPERATORS

Each task is parsed into an open-vocabulary TaskContract

$$\mathcal{T}_\tau = (A_\tau, A_\tau^{src}, A_\tau^{sink}, V_\tau, O_\tau, K_\tau, C_\tau, F_\tau, P_\tau), \tag{1}$$

where A denotes applications, A^{src} and A^{sink} source and destination roles, V actions, O objects, K key terms, C constraints, F finish gates, and P operational subproblems. The contract is independent of benchmark task names and supplies shared operational semantics for experience scope and retrieval permission.

An experience item is

$$m = (\mathrm{id}, t, s, z, b, p), \tag{2}$$

where t is its functional type, s its transfer scope, z its contract tags, b its actionable content, and p its provenance. The item is interpreted as a conditional operational hypothesis: when a task satisfies scope s and tags z, supplying b may improve the behavior associated with type t. Workflows, bridges, answer-extraction rules, UI locators, and warnings describe different behavioral roles; contract, subproblem, application, and global scopes describe different transfer ranges.

The generator proposes a bounded atomic edit to candidate memory:

$$e = (\mathrm{op}, m, \mathrm{target}, y_{src}, s_{prop}, p), \quad \mathrm{op} \in \{\texttt{add}, \texttt{replace}, \texttt{delete}\}. \tag{3}$$

An `add` creates a hypothesis, a `replace` revises a specific incumbent, and a `delete` proposes retiring one. Each edit records source outcome, proposed scope, target, and evidence. A `replace` or `delete` points to a specific incumbent, which remains available as a comparator and rollback point until promotion applies the edit.

Source outcome acts as a conservative prior over generation permissions. The default policy lets successful traces propose positive procedures and restricts failed traces to failure mechanisms, counterexamples, or warnings. Removing this condition also permits positive recovery strategies from failures. Regardless of source, generated items remain candidates until promotion.

Semantic consolidation merges evidence and provenance from exact duplicates, aliases, and compatible near-duplicates into a canonical item. Candidate and incumbent versions coexist for replace operations, and delete targets remain available until a decision is applied. Scope has contract, subproblem, application, and global levels. A broad scope requires compatible evidence from distinct tasks; otherwise the candidate remains tied to its source contract or subproblem.

H.2 Exposure and Deployed-Memory Updates

For each task–memory pair, the exposure variable is

$$q_{\tau,m} \in \{\text{TRIAL}, \text{CONTROL}, \text{DEPLOYED}, \text{NONE}\}. \tag{4}$$

Here, TRIAL means that the candidate enters the prompt for evidence collection; CONTROL denotes a matched episode without the candidate treatment; DEPLOYED means that the item was promoted before the episode and enters through normal reuse; and NONE means that the item does not influence execution. Trial exposure does not itself change deployment state.

Only promoted edits change deployed memory:

$$M_{v+1} = \text{Consolidate}(\text{Apply}(M_v, e), s_e), \qquad P(e) = 1. \tag{5}$$

Other candidates remain pending, rejected, or shadow. Consolidation absorbs redundant updates into a canonical item, while scope calibration narrows transfer when cross-task evidence is absent. Promotion applies only at a pass boundary.

H.3 Task Contract and Retrieval Details

Textual relevance first produces a high-recall set $\mathcal{N}(x_\tau; C \cup M)$. Experience tags are then matched field by field against the TaskContract and classified as matched, partial, missing, or conflicting. Matched and partially matched tags form weighted coverage, while conflicting tags form a weighted penalty:

$$\text{cov}(m, \tau) = \sum_{z \in M} w_z + \rho \sum_{z \in P} w_z, \qquad \text{conf}(m, \tau) = \sum_{z \in X} w_z. \tag{6}$$

Critical conflicts such as import/export, copy/delete, a wrong recipient, opposite polarity, or mutually exclusive finish gates form a hard veto that reranking cannot override.

The feasible retrieval set additionally requires compatible scope, no critical conflict, and sufficient contract coverage:

$$\mathcal{F}(\tau) = \{m \in \mathcal{N}(x_\tau; C \cup M) : \text{scope}(m, \tau) = 1, \quad \text{critconf}(m, \mathcal{T}_\tau) = 0, \quad \text{cov}(m, \mathcal{T}_\tau) \geq \theta\}. \tag{7}$$

Similarity determines which experience to inspect, not which experience is authorized for context injection. Trial retrieval is restricted to candidates assigned to the current episode, whereas deployed-memory retrieval is restricted to items promoted before execution. The retriever chooses at most one primary workflow and one complementary warning, and abstains when no compatible item remains.

I Implementation and Reproducibility

I.1 Agent and Environment Integration

The framework connects to an existing agent through `before-task` and `after-task` boundaries. Before execution, it builds a TaskContract from the instruction and environment metadata, retrieves deployed memory, and, when assigned, exposes a candidate through the trial channel. It appends the resulting context through the agent's existing prompt or context channel. The agent's predict, step, evaluate, and teardown loop remains unchanged. After execution, the framework records task score, action trace, actual exposure, and finish conditions as a normalized episode artifact.

An environment adapter supplies metadata for contract construction, automatic scoring, and procedures for locating behavioral evidence. Experience representation, state transitions, and retrieval policy remain shared across environments. Benchmark-specific adapters do not change the agent policy, action schema, or environment transitions.

I.2 Trial Assignment and Evidence Records

For later compatible task–candidate pairs, a stable hash and pass index assign trial or control conditions and alternate them over passes. A workflow compares incumbent+candidate against incumbent; a warning keeps the same primary and changes only the warning; and replace/delete compares applying the edit against retaining the incumbent. Estimation retains only task strata observed under both conditions.

Each evidence record states whether the candidate was selected, whether it reached the per-step prompt, task score, trajectory steps exercising the claimed subproblem, finish gates, snapshot hash, and a unique ID. Episodes are excluded when the candidate was not injected, the scorer was unavailable, snapshot semantics differed, or relevant behavior could not be located.

I.3 Snapshots, Concurrency, and Recovery

Each learning pass pins a read-only snapshot. Episodes may read that snapshot concurrently and write independent backlogs; consolidation, promotion, and publication commit at the pass boundary. An EvolutionLedger records candidates, deduplicated evidence, decisions, comparators, and checkpoint hashes. A run manifest pins the code revision, models, image, task order, arguments, and pre/post store hashes.

Checkpoints make pass-boundary state immutable and allow compatible resume after transient infrastructure failures. Resume rejects configuration, identity, task-order, store, or manifest mismatches. Retries do not alter the frozen task definition, model, image, memory snapshot, or selection decision, and invalid infrastructure attempts are excluded from valid episode totals.

I.4 Token and Memory Accounting

Experience-context prompt tokens are measured from execution-model `usage.prompt_tokens` differences rather than estimated from characters. Materialized deployed-memory size is counted from immutable checkpoint files rather than ledger decisions. The same accounting distinguishes actual candidate-trial context, deployed-memory reuse, and episodes with no experience context.

I.5 Memory-System Overhead

The TaskContract is constructed deterministically and consumes zero model tokens. Table 17 reports the pass-wise candidate-record bytes, vector-index bytes, deployed-memory bytes and items, and retrieved-context tokens available from the frozen historical R1 audit.

Table 17: Memory-system overhead in the audited historical R1 STEER run. Byte counts are exact serialized filesystem measurements; context tokens are execution-service usage counts.

Pass	Candidate bytes	Index bytes	Items	Deployed bytes	Context tokens
1	473,529	1,196,996	0	0	0
2	880,649	2,191,223	0	0	13,929
3	1,287,329	3,083,804	0	0	21,440
4	1,655,433	3,865,087	0	0	19,250
5	2,050,958	4,727,568	1	10,451	28,743

Candidate storage is not bounded: by Pass 5, serialized candidate records occupy 2,050,958 bytes and their vector index occupies 4,727,568 bytes, while the deployed store contains one 10,451-byte item. Complete historical generation-token accounting, some model-compatibility cost, and reliable component-level latency are unavailable. We therefore make no claim about total system compute or negligible storage overhead.

I.6 Frozen Experimental Records

All MW30 conditions use the same task-order hash and empty-store hash. Frozen experiments record code revision, models, image, task membership, parameters, episode validity, and pre/post

knowledge-base hashes. Resume rejects configuration or identity mismatches, and final store hashes for every frozen retrieval condition equal the initial hash. The final experiment code passes 179 tests.

The MobileWorld anchor comes from complete 117-task trajectories, while mechanism experiments use independent matched MW30 controls. The targeted A1 study contains 24 valid episodes; each fixed-candidate A2 policy contains 80; the A2 promoted/rejected probe contains 144; and each Retrieval24 condition contains 72. Final reports contain no unresolved invalid episode.

I.7 Main-Study Validity and Frozen Identity

The MobileWorld manifest records the task definitions and order, execution and knowledge-model identities, container image, scorer, memory-store snapshot, and all associated hashes before execution. An episode is valid only when environment initialization succeeds, the specified task and snapshot are used, the automatic scorer is available, and the complete result and treatment audit are written. Infrastructure failures are retried without changing the task, model, image, scorer, store, or selection decision; failed infrastructure attempts are excluded from valid episode totals. The historical NoMemory task-order mismatch and its implications are reported with the ExactToken audit in Appendix B.1.

I.8 Exposure-Matched and Targeted Protocols

Count-Matched Similarity retains the append-all accepted-memory store and similarity top-2 retriever. Before each pass, a deterministic selection procedure freezes exactly 0, 24, 36, 33, and 48 selected task episodes, respectively, in a read-only manifest. The matching constraint concerns the number of selected episodes in each pass, not the identity of every selected task. The manifest, store snapshot, ranking output, and treatment audit are fixed before the first episode of the pass.

The targeted studies freeze the candidate or promoted item, task snapshot, execution model, prompt template, scorer, and read-only memory state. Each predefined task or task–memory pair receives three treatment repeats and three matched controls. A treatment-integrity record verifies whether the intended item actually reached the prompt; invalid or mismatched exposures are excluded. Candidate trials compare forced candidate inclusion with withholding that candidate, while the post-promotion study compares ordinary deployed-memory reuse with a no-memory control.

Table 18 retains the aggregate gain/loss matrices; the complete predefined task and task–memory rows used by Figure 8 are preserved in the frozen source CSVs.

Table 18: Aggregate paired and targeted outcome matrices for the exposure controls.

Comparison	Positive/gain	Negative/loss	Unchanged
STEER vs. Count-Matched Similarity	16	3	98
STEER vs. ExactToken Similarity	10	0	107
Candidate trial vs. control	5	1	3
Deployed reuse vs. no memory	4	0	6

I.9 Complete OSWorld V2 Setup

OSWorld V2 contains 108 desktop tasks. The execution agent is Seed-2.0-Pro, and Base and STEER are evaluated for five passes. Partial Reward measures progress toward the task objective, whereas Binary Reward requires strict completion. The experiment starts with an empty environment-specific memory state and does not transfer MobileWorld memory items into OSWorld.

J Visual Mechanisms and Targeted Memory-Use Studies

Figure 13 traces the candidate's two trials and two controls before admission. Figure 14 shows how the shared operational representation determines permission to inject a recalled item.

Table 19: OSWorld V2 results on 108 tasks. Cumulative partial reward averages each task's best score across passes; strict completions count unique fully solved tasks.

	Attempt partial (%)		Cumulative partial (%)		Strict completions	
Pass	Base	STEER	Base	STEER	Base	STEER
1	7.90%	6.46%	7.90%	6.46%	0/108	0/108
2	6.78%	7.28%	9.53%	9.07%	0/108	0/108
3	7.35%	8.12%	11.35%	12.01%	0/108	0/108
4	8.10%	**9.56%**	12.88%	15.47%	0/108	1/108
5	7.62%	7.18%	13.69%	**16.35%**	0/108	**2/108**

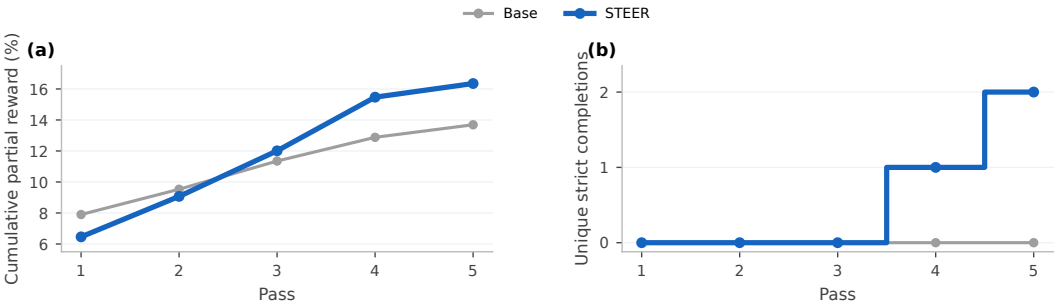

Figure 12: Independent evolution on OSWorld V2 from an empty desktop memory store. (a) Cumulative partial reward increases across five passes. (b) Strict completions emerge for STEER in later passes.

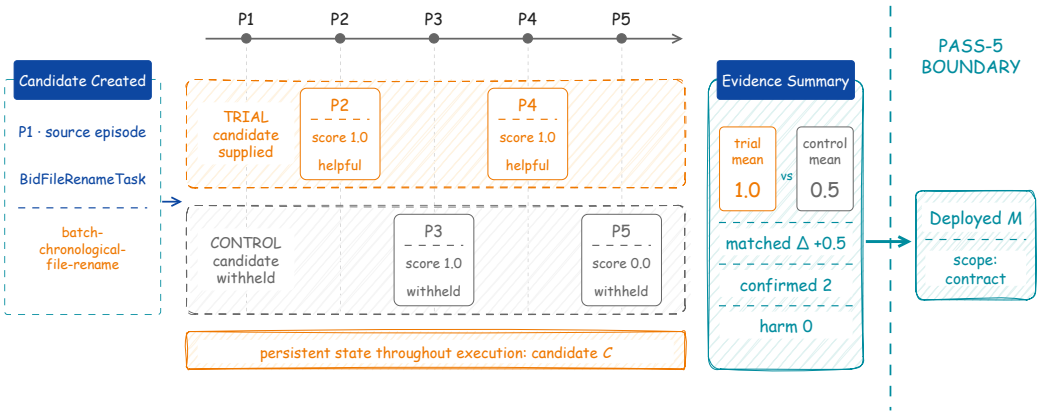

Figure 13: Trial before deployment for the frozen *batch-chronological-file-rename* item. Created from *BidFileRenameTask* in Pass 1, it receives two candidate trials and two controls before contract-scoped admission at the Pass-5 boundary.

The identity-matched replay includes 141 frozen task-pass identities and two repetitions per condition. The 32 cross-task identities are a diagnostic subset. Figure 15 retains the complete selection visualization.

For nine predefined candidate-task comparisons, each arm has three repetitions: five task-level differences are positive, three neutral, and one negative. The mean difference is $+0.333$ (95% CI $[+0.074, +0.593]$, exact sign-flip $p = 0.094$). For ten deployed task–memory pairs with three repetitions per arm, four differences are positive and six neutral, with mean $+0.20$ (95% CI $[+0.033, +0.400]$, sign-flip $p = 0.125$). These selected-pair interventions do not estimate a benchmark-wide effect.

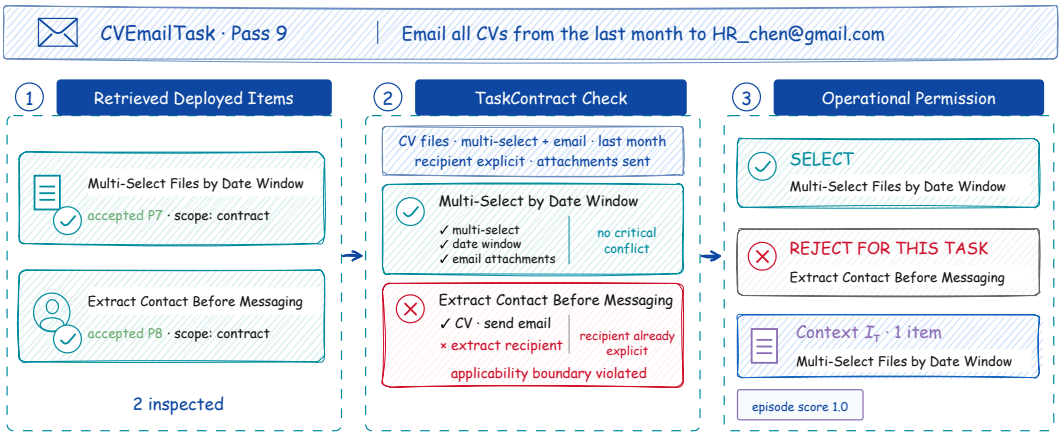

Figure 14: Operational permission in Pass-9 `CVEmailTask`. Recall proposes two deployed items. Date-window multi-selection matches the file and attachment operation, whereas recipient extraction is unnecessary because the recipient is explicit. Only the compatible procedure enters the context.

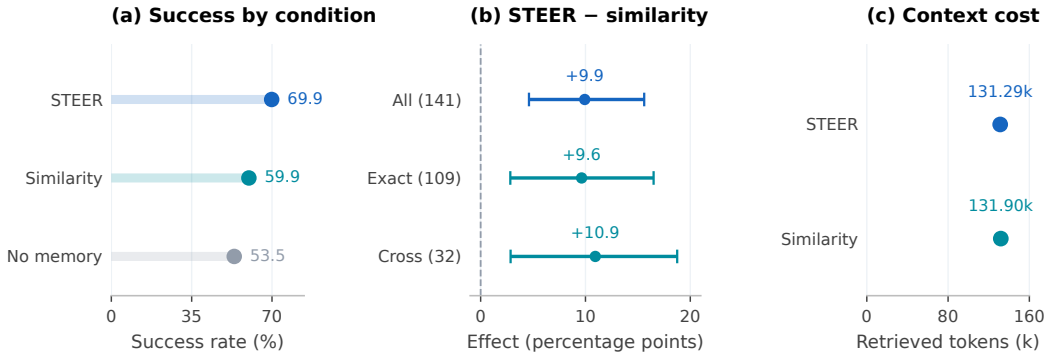

Figure 15: Identity-matched selection: condition means, STEER-minus-similarity effects overall and by provenance, and the nearly identical context costs of the memory conditions.

Table 20: Source-disjoint stress test: experience originating from the current task is excluded from all memory-use paths.

Method	P@1	P@2	P@3	P@4	P@5	Memory-injected episodes	Context tokens
SD-Similarity	64	78	83	84	88	468	294,087
SD-STEER	66	78	81	87	90	60	40,100

The longer R1 trace is reconstructed from the per-pass MobileWorld task reports and the admission ledger preserved before Pass 10. Each of Passes 1–9 contains 117 task results. The available Pass-10 report contains 87 results and is excluded from the complete-pass figure. Accepted and pending counts use post-pass decisions, rather than pre-admission item-file states. The ledger records no applied replacements or deletions over these nine complete passes.

Model marks in Figure 1 are adapted from Lobe Icons (`https://github.com/lobehub/lobe-icons`, MIT license); the ByteDance mark identifies the Seed executor. Public-reference snapshots and plotted values are preserved with the figure sources.

K Scope of the Evidence

Scope and limitations. Our scaling evidence concerns deployed memory and injected context over the tested horizon. Candidates continue to accumulate; the nine-pass trace establishes nei-

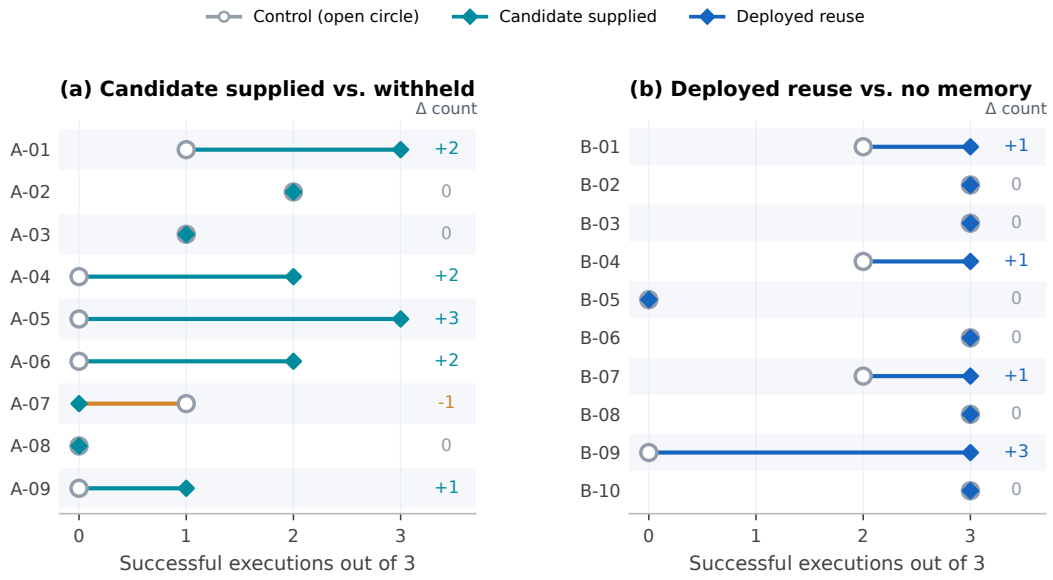

Figure 16: Per-pair outcomes behind Figure 8. All nine candidate-task comparisons and ten deployed task–memory pairs are retained, with three executions per arm; overlapping markers indicate ties.

ther bounded total storage nor sustained replacement/deletion effects. Context savings exclude full generation cost. The outcome-conditioning ablation does not isolate a success-rate gain from representation, and admission is not a universal knowledge-quality test. Desktop results come from one independently evolving comparison; targeted reuse studies cover selected pairs.