HURRICANE: ORGANIZING EXECUTION EXPERIENCE WITH EVIDENCE AND REUSE CONDITIONS

Anonymous authors
Research manuscript

ABSTRACT

A useful agent memory must retain the evidence and conditions that justify reusing past experience. We introduce Hurricane, which binds reusable content to source evidence, applicability conditions, dependencies, and public validation receipts. It separates retrieval relevance from reuse eligibility: an API admission gate checks retrieved records against current public state before granting access. Controlled studies examine organization, placement, and admission. Structured experience leads on HotpotQA, whereas a prose rendering of the same evidence leads on 2Wiki. Following Trace as State, prepending organized same-task state improves mean Parents exact match over appending by 16.8–25.4 percentage points across models; BFS exhibits mixed position preferences. In a constructed 36-target API cohort, Hurricane was observed to achieve the same success count as Hybrid (32/36 each) with 58.5% fewer known tokens including allocated source preparation. This single-run result is specific to that cohort. Across studies, organization, placement, and admission have distinct, task-dependent effects.

1 INTRODUCTION

Language-model agents solve tasks through sequences of decisions, tool calls, and feedback from an external environment (Yao et al., 2023; Yang et al., 2024). These interactions provide experience that can inform later decisions without updating model weights. The challenge is that experience is conditional: a procedure succeeds under particular requirements, dependencies, and environmental conditions. A numeric-conversion repair that assumes half-up rounding, for example, may be incorrect when a later task requires half-even rounding. Storing the repair preserves what worked, but may lose why it worked or when it applies. Effective reuse therefore depends on retaining the connection between a procedure, its supporting observations, and its conditions of use.

Recent work has developed complementary ways to capture and organize this experience. Reflections, general insights, and executable skills preserve lessons from interaction (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024), while memory architectures manage context and link stored records (Packer et al., 2023; Xu et al., 2025). Feedback and verification already play important roles in these systems. Even a record that passed checks during acquisition may become unsuitable when dependencies or requirements change. This motivates a separate decision at reuse time: establishing which of a candidate's declared conditions still hold. Retrieval relevance and reuse eligibility address different questions, and a high retrieval score alone cannot resolve the latter.

Hurricane models experience reuse as a contract-checking problem. Its record schema binds reusable content to source evidence, applicability conditions, dependencies, and public validation receipts. Retrieval identifies relevant candidates; an admission gate checks their eligibility against current public state before granting access. A changed dependency or stale validation receipt can therefore prevent a highly ranked record from reaching the agent. Source links identify the observations behind the record, while receipts record what was checked and under which state. Separating retrieval from admission makes the grounds for reuse inspectable and allows admission policies to be compared under a fixed retrieval ranking.

The evaluation isolates organization, placement, and admission through controlled studies of *cross-task experience* and unverified *same-task state*. QA tests organization without admission, including a prose control that holds evidence content fixed. Following Trace as State (Zou & Tang, 2026), long-context tasks hold organized state fixed while varying its input position. Constructed API

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.

tasks evaluate the gate under shared retrieval rankings and execution budgets. Across these studies, representation and position preferences depend on the task, while selective admission yields token savings with mixed effects on success.

2 Related Work

Agent memory and experience learning. Recent work increasingly learns how agents manage memory. AgeMem trains a policy over long- and short-term memory operations (Yu et al., 2026), while MemSkill learns and evolves routines for extracting and revising memories (Zhang et al., 2026a). Dynamic Cheatsheet and ACE maintain evolving contexts through self-curation and incremental updates (Suzgun et al., 2026; Zhang et al., 2026b). These approaches build on experience retention through reflections, insights, and executable skills (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024), and on reusable workflows and case memory (Wang et al., 2025; Zhou et al., 2025). A-MEM, Mem0, and G-Memory organize linked or consolidated records (Xu et al., 2025; Chhikara et al., 2025; Zhang et al., 2025); memory hierarchies and forgetting govern their availability (Packer et al., 2023; Kang et al., 2025; Zhong et al., 2024). These developments extend the role of memory in planning and control (Sumers et al., 2024; Park et al., 2023). Hurricane focuses on a decision that remains distinct from learning what to store: whether a record's declared requirements hold in the current environment. Our frozen cross-task archives isolate admission from improvements due to memory accumulation or evolution.

Evidence selection and conditional reuse. Retrieval-augmented generation supplies external evidence (Lewis et al., 2020), with hierarchical and adaptive retrieval extending how it is selected (Sarthi et al., 2024; Jiang et al., 2023). Self-RAG and CRAG evaluate retrieved evidence (Asai et al., 2023; Yan et al., 2024); RECOMP and LongLLMLingua compress or select context (Xu et al., 2024; Jiang et al., 2024). MemRL further separates semantic recall from selection using learned utility estimates (Zhang et al., 2026c). This separates relevance from expected usefulness; Hurricane separates relevance from eligibility under a declared contract. A record with high estimated utility can still have changed dependencies or an expired validation receipt. The gate therefore checks these requirements against current public state before allowing the agent to access the record. This design draws on provenance, software contracts, and runtime verification (Green et al., 2007; Meyer, 1992; Leucker & Schallhart, 2009). Source links identify supporting observations; receipts bind public checks to the record and state they concern. Hurricane uses these bindings to govern access under a fixed retrieval ranking. The resulting access decision is auditable against the record's declared contract.

Memory use and state placement. Memory evaluation spans conversational recall (Maharana et al., 2024; Wu et al., 2025), long-context reasoning (Bai et al., 2024; Hsieh et al., 2024), and, increasingly, interdependent agentic tasks. MemoryArena evaluates memory through multi-session interaction with an environment (He et al., 2026). Hurricane's component studies isolate organization, admission, and placement; they do not evaluate such an online memory loop. Position sensitivity (Liu et al., 2024) motivates our use of the Trace as State intervention (Zou & Tang, 2026), holding organized, unverified same-task state fixed between positions. This differs from revising an answer through Self-Refine (Madaan et al., 2023). Experience faithfulness asks whether supplied history causally affects behavior (Zhao et al., 2026); our input audits establish what was delivered, without by themselves establishing that causal dependence.

3 Organizing and Reusing Execution Experience

Hurricane separates what a record preserves, whether it is eligible for access, and how it enters the agent's context. The API and same-task paths share evidence-linked organization but differ in delivery (Figure 1).

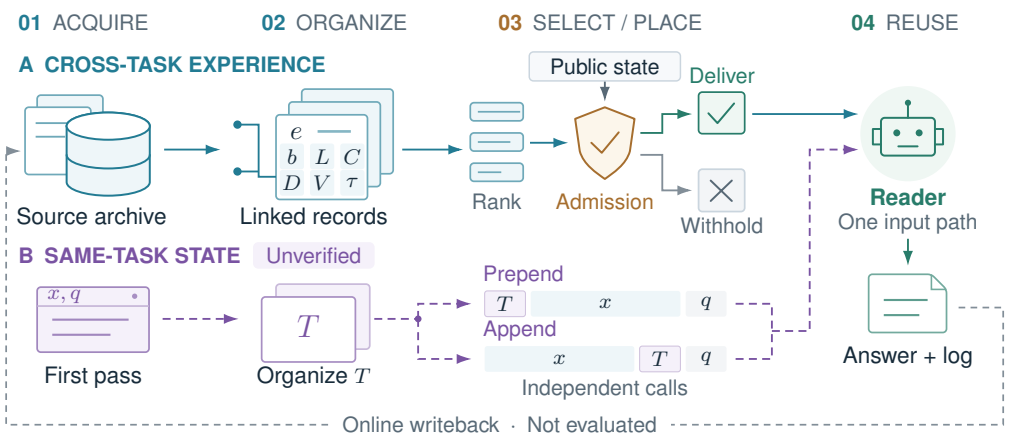


Figure 1: API admission and long-context state delivery. API records are checked against current public state; unverified T is shared between position conditions. QA evaluates organization without the gate. Online writeback is unevaluated.

3.1 Evidence-linked records

An organizer G converts source executions z_i into records $E_i = G(z_i)$, forming the cross-task archive $\mathcal{E} = \bigcup_i E_i$. Each record $e = (b, L, C, D, V, \tau)$ has six fields:

Field	Meaning	Contents
b	Reusable content	Observations, procedures, or an implementation.
L	Source links	References connecting the content to source evidence.
C	Applicability	Explicit conditions under which the content applies.
D	Dependencies	Required inputs or artifacts and their recorded bindings.
V	Validation receipts	Results of public checks tied to a contract and state.
τ	Optional detail	Additional observations or execution-trace detail.

In the rounding example, b holds code, C specifies its supported mode, and D binds its configuration file. L locates supporting observations; V records public check results under that configuration.

The shared record interface leaves source extraction, required fields, checks, and rendering to task adapters. Source admission checks structure and evidence links before storage. Fields not required by the adapter may be empty, and τ is optional. Missing evidence cannot satisfy a required check. QA evaluates this organization interface without a reuse-time gate; admission is evaluated only in the API suite.

3.2 Contract-based admission for cross-task reuse

Retrieval ranks records by relevance; admission checks whether their declared requirements hold in the current environment. Hurricane expresses these requirements as an API contract binding record identity and content to applicability conditions, dependencies, and public checks. A highly ranked half-up procedure can therefore be withheld from a half-even target.

Let s denote current public facts and artifact bindings, and $\mathcal{R}$ the host-supplied receipts. Eligibility requires five predicates to pass:

$$a(e, s, \mathcal{R}) = \bigwedge_{j=1}^{5} \chi_j(e, s, \mathcal{R}). \tag{1}$$

In order, the predicates check record bindings, task scope, applicability, transitive dependency closure, and receipt validity. Each returns one if satisfied or not required, and zero if unmet or unresolved. Valid receipts match the check definition, record contract, and current state, remain unexpired, and report a passing check with at least one executed test.

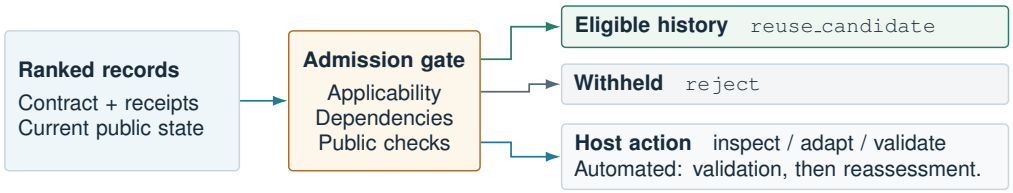


Figure 2: Admission after retrieval. Eligible records become available as history. Other routes withhold access and explain why, or identify the host action needed for reassessment.

The gate returns a route and its reasons (Figure 2). When $a = 1$, `reuse_candidate` permits historical access. Otherwise, the failed requirement determines the next action: `inspect` for unresolved evidence or bindings, `adapt` for changed dependencies with continued applicability, `validate` for missing or invalid receipts, or `reject` for incompatible scope or conditions. The evaluated host automates only validation: it uses a qualifying cached receipt or runs an isolated public check, then reassesses the record and restores the target workspace. Admission grants access to history; it does not install code or establish task success (Appendix J.2).

For task context x and request q, the API studies fix a shared Hybrid ranking R, combining lexical and vector rankings by reciprocal rank fusion (Cormack et al., 2009). After any validation and reassessment, Hurricane retains eligible records in their original order and greedily packs whole records into the initial history message M:

$$R = \mathrm{Rank}_{\mathrm{Hybrid}}(x, q, \mathcal{E}), \qquad R^{+} = \langle e \in R \mid a(e, s, \mathcal{R}) = 1 \rangle, \tag{2}$$

$$M = \mathrm{Pack}_{B}(R^{+}), \qquad \ell(M) \le B, \quad \mathrm{IDs}(M) \subseteq \mathrm{IDs}(R^{+}). \tag{3}$$

Here B is the initial-history token cap, ℓ counts tokens, and IDs lists included record identities. The cap and subsequent access policy are distinct: in the main 36-task comparison, Hurricane's history tools can access all admitted records in R^{+}; in the twelve-task comparison, every condition restricts those tools to the records selected for its initial pack. Tool returns count toward the target execution budget. Neither organization nor admission receives hidden answers or private tests.

3.3 Controlled placement of same-task state

Same-task reuse carries information between attempts on one question. From the first execution $z^{(0)}$, the long-context adapter organizes the returned reasoning field into $T = \mathrm{Render}(G(z^{(0)}))$. The separate answer field is excluded; missing reasoning fails acquisition. T **is unverified model-generated state.** Quotation matching links its entries to the source trace without establishing semantic correctness, and that trace may itself contain attempted answers.

To isolate input position, two fresh solver calls receive the same T, task context x, and question q. Following Zou & Tang (2026), their user messages are

$$P_{\mathrm{pre}} = [T, x, q], \qquad P_{\mathrm{post}} = [x, T, q]. \tag{4}$$

Only the order within the user message changes: q remains last, and the system instruction and solver settings are identical. Neither branch receives the other's output. First pass and organization are rerun per question and repeat; only that repeat's two position branches share T.

4 Experiments

4.1 Experimental setup

The studies ask three component-level questions: what evidence to organize (QA), where to place same-task state (long contexts), and which retrieved records to admit (API). Each study fixes the relevant source, state, or retrieval ranking; these comparisons do not estimate joint effects or end-to-end advantage over complete systems such as A-MEM, Reflexion, or ExpeL. Unless specified, experiments use SiliconFlow's DeepSeek-V3; Appendix A gives settings and protocols.

QA reports answer exact match (EM) and F1 on HotpotQA (Yang et al., 2018) and 2WikiMultihopQA (Ho et al., 2020). We report node-set EM and F1 on the breadth-first search (BFS) and Parents tasks in GraphWalks (OpenAI, 2026). On MRCR v2 (Vodrahalli et al., 2024), we report full-response EM alongside SequenceMatcher similarity (Seq.) (Google DeepMind, 2026). For API tasks, a solution succeeds only if the agent submits an artifact that passes all private contract checks. Passing a public check during execution does not count as task success.

4.2 Organizing experience for question answering

The complete QA experiment tests the record schema through organization and presentation controls. **It does not invoke the admission gate.** Reuse-time admission is evaluated separately, only in the API suite.

Table 1 compares five ways of presenting source experience on HotpotQA and 2Wiki, with one execution per target and condition.

Raw log preserves observable execution history; *Prose summary* and *Hurricane structured* use separate model calls to organize the same source. These conditions can differ in both the information retained and its presentation. *Raw log + extra budget* reallocates the organizer's allowance to solving, providing a control for additional computation (Appendix A.1).

Holding evidence fixed. *Same-evidence prose* is generated deterministically from the Structured record, without another model call. It preserves fact text, evidence identifiers, categories, entry order, and source identifiers. If either representation exceeds the history budget, both drop the same entries. This control separates evidence selection from its presentation to the reader; wording and token length can still differ.

	HotpotQA		2Wiki	
Condition	EM (%)	F1 (%)	EM (%)	F1 (%)
Raw log	44.1	56.1	51.5	52.8
Prose summary	44.1	58.3	57.6	58.1
Hurricane structured	**49.0**	**63.5**	57.6	58.8
Raw log + extra budget	47.1	60.1	53.0	54.3
Same-evidence prose	45.1	59.0	**59.1**	**61.3**

Table 1: Results of the complete QA experiment. Conditions share target questions and source executions, with one run per target and condition. All targets, including failures, contribute to EM and F1 (percentages). The best score in each column is bolded.

Task performance. Structured leads on HotpotQA (49.0% EM, 63.5% F1), exceeding same-evidence prose by 3.9 and 4.5 percentage points. On 2Wiki, it ties Summary's EM but trails same-evidence prose (59.1% EM, 61.3% F1). This reversed contrast shows that preserving evidence does not determine the best presentation. Raw + budget reaches 47.1% and 53.0% EM, respectively, falling short of the best score on either dataset despite its additional solving allowance.

4.3 Placing same-task state in long contexts

We evaluate DeepSeek-V4-Flash, Qwen 3.7 Max, and GLM-5.2 on GraphWalks-256K BFS and Parents (200 questions per task) and MRCR v2 with eight needles (200 questions each at 256K and 512K). MRCR scores pool both length groups. Table 2 reports means over five runs of the full pipeline. All samples contribute to the results, with timed-out requests retried.

Each model regenerates First pass and T for every question and repeat, then runs both position branches (Equation 4). Traces retain at most their first 50,000 characters before organization; Appendix A.2 specifies settings and scoring.

What the placement comparison identifies. Prepend and Append share the same organized T within each repeat, so their contrast isolates input position. It does not establish whether structured

state outperforms raw traces or prose summaries. That raw/prose/structured comparison was blocked by pipeline acquisition and request-delivery failures (Appendix H).

Condition	BFS EM / F1	Parents EM / F1	MRCR EM / Seq.
DeepSeek-V4-Flash			
First pass	81.3 / 83.3	78.3 / 91.3	62.5 / 71.5
Hurricane Prepend	90.8 / 91.7	**99.6 / 99.7**	**75.0 / 83.1**
Hurricane Append	**95.1 / 95.8**	78.3 / 88.4	**75.0** / 75.6
Qwen 3.7 Max			
First pass	60.1 / 68.2	61.0 / 87.3	79.6 / 83.0
Hurricane Prepend	**64.0 / 70.9**	**96.4 / 99.1**	**88.2 / 91.2**
Hurricane Append	60.6 / 70.5	71.0 / 91.7	83.8 / 87.1
GLM-5.2			
First pass	55.6 / 70.4	66.6 / 88.5	40.2 / 48.2
Hurricane Prepend	**63.4** / 75.0	**99.8 / 99.8**	**61.2 / 71.9**
Hurricane Append	60.6 / **76.0**	83.0 / 92.5	39.8 / 58.1

Table 2: Long-context performance, averaged over five runs of the full pipeline for each question. Each repeat generates a new T shared by Prepend and Append. BFS and Parents each use 200 questions; MRCR pools 200 cases at each of 256K and 512K tokens. Scores are percentages; Seq. denotes SequenceMatcher similarity. The best score per model and metric is bolded, including ties.

Mean Parents EM and F1 favor Prepend for all three models. EM gains over Append are 21.3, 25.4, and 16.8 percentage points for DeepSeek, Qwen, and GLM, respectively.

Position preference depends on the task and model. BFS EM favors Append for DeepSeek by 4.3 points, but Prepend for Qwen and GLM by 3.4 and 2.8 points. GLM's BFS F1 favors Append (76.0 versus 75.0). MRCR EM is tied for DeepSeek and favors Prepend by 4.4 points for Qwen and 21.4 for GLM; Seq. favors Prepend throughout. Figure 3 summarizes the mean EM contrasts.

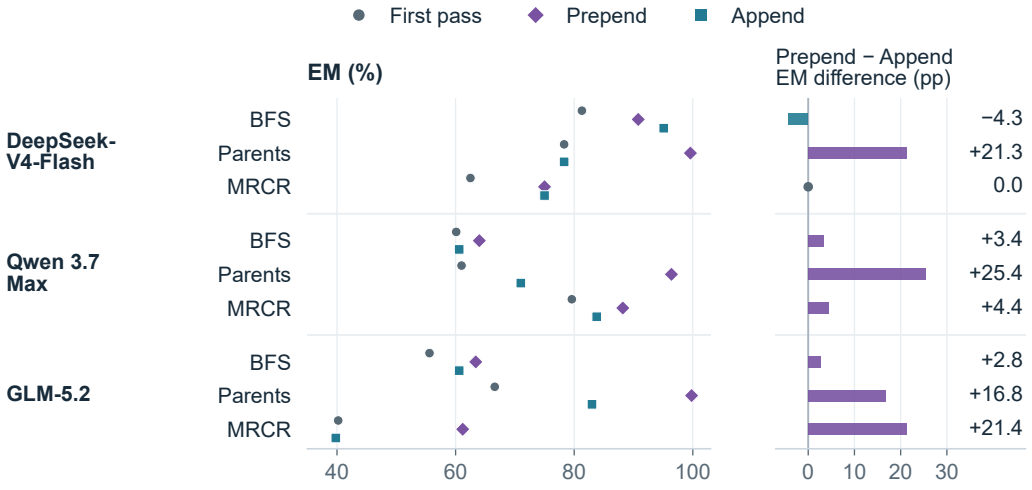


Figure 3: Effect of state placement on long-context accuracy. Left: mean EM over five runs of the full pipeline. Right: Prepend minus Append in percentage points, calculated from the displayed means. Prepend improves mean Parents EM for all three models; the preferred position on BFS depends on the model.

Compared with First pass, Prepend improves mean EM on all nine model–task combinations. This comparison changes computation as well as context: both second passes use a first execution and an organizer. Earlier placement studies are reported separately in Appendix C.

4.4 SELECTIVE REUSE: TASK SUCCESS AND TOKEN USE

The constructed API tasks require the agent to implement and submit a Python `solve(value)` function from a public contract, options, and examples. The main comparison crosses six families with six reuse scenarios, giving 36 targets (Appendix B). Separate comparisons examine six-task efficiency and an expanded twelve-task mechanism cohort.

Condition	Tasks given initial history	Tasks solved	Known model tokens	
			Target	Source + target
Hybrid	36/36	32/36	451,520	493,638
Hurricane	15/36	32/36	162,539	204,657
Ordinary log + same checks	15/36	30/36	183,914	226,032
No history / no extra feedback	0/36	30/36	228,481	228,481
Self-Refine adaptation	0/36	24/36	233,378†	233,378†

Table 3: API results on 36 tasks (SWE-Agent, DeepSeek-V3; one run per task and condition). Initial history counts targets receiving a record; success requires submission and all private checks passing. Tokens are summed over targets; Source + target adds 42,118 preparation tokens per memory condition. Tool costs are excluded. †One Self-Refine call has unknown usage, excluded from its subtotal.

Baselines and shared budget. The 36-task memory conditions share six frozen sources. *Hybrid* exposes the full ranked archive; *Hurricane* filters it through dependency, applicability, and validation checks. In the mechanism comparison, *Hybrid top-1* limits initial history and later history tools to the highest-ranked record. *Ordinary log + same checks* shares Hurricane's admission and serialization. *No history* receives the task and tool feedback; *Self-Refine* adds up to two same-model feedback and revision rounds within budget, without cross-task history.

Each condition runs once per target (180 executions), using SWE-Agent and DeepSeek-V3 with aligned public inputs and workspaces. Limits are eight model calls, 160,000 tokens, and 600 tool seconds, including feedback; initial history is capped at 4,096 tokens. All targets, including failures, remain in scoring. Outcomes describe these constructed instances.

Success and token use. In this 36-target cohort, Hurricane and Hybrid each solved 32/36. Hurricane used 64.0% fewer known target tokens: 162,539 versus 451,520 (Table 3). Including the same 42,118-token source allocation gives 204,657 versus 493,638 known tokens, a 58.5% reduction. These savings accompany selective history delivery: Hurricane supplies initial history on 41.7% of targets, compared with Hybrid's 100.0%.

What the token figures include. Target-only tokens cover target model calls, including inputs containing tool returns; source-plus-target adds allocated preparation. Both include known usage from failed calls and leave missing usage unresolved. Programmatic admission checks consume target tool time but make no model calls. Token totals exclude tool runtime and infrastructure overhead; they do not measure total system cost.

What admission changes. Admission retains the leading record on 15 targets and withholds history on 21. Hurricane and Hybrid both solve all 15 in the first group and 17/21 in the second, with two wins and two losses in the latter (Appendix K). The matched log produces the same initial inputs as Hurricane and solves 30/36. Since the initial representations are identical, this difference cannot be attributed to their format.

No established token advantage over top-1. At the 4,096-token history cap and shared target allowance, Hurricane, Full Hybrid, top-1, the matched log, and no validation all solve 6/6 (Appendix G). Hurricane uses 3,363 mean target tokens versus 13,306 for Full Hybrid, a 74.7% reduction. Against top-1's 3,751, the observed reduction is 388 tokens (10.3%). The 95% paired bootstrap confidence interval for the mean target-token difference (Hurricane minus top-1) is $[-1{,}073.5,\ 250.0]$ tokens

per task. **Because this interval includes zero, this six-task comparison cannot establish that Hurricane uses fewer tokens than top-1.**

Condition	History policy	Success (%)
No history	No historical experience	41.7
Hybrid top-1	Highest-ranked historical record	75.0
Hurricane w/o validation	Other admission rules retained; validation check removed	75.0
Hurricane (full)	Admission based on applicability, dependencies, and validation	**100.0**

Table 4: Observed success on twelve constructed API tasks, with one run per task and condition, a shared target budget, and a 4,096-token cap on initial history. Success requires all private checks to pass. These exploratory rates do not establish deployment performance (Appendix G.1).

Exploratory mechanism cohort. The exploratory twelve-task constructed cohort extends the six-task efficiency cohort. Full admission was observed to achieve 12/12 private-check passes versus 9/12 for both top-1 and no validation, a 25-point difference; no history gives 41.7% (Table 4). Conditions share the model, target budgets, and 4,096-token history cap, with one run per task and condition. These counts do not estimate performance on natural workloads. In this cohort, every condition restricts history tools to its initial selection; top-1 selects at most one record. The token estimates above cover only the original six tasks; the expanded protocol is in Appendix G.1.

How the checks change reuse. A separate direct-execution cohort yields 31/36 successes for Hurricane and 28/36 for Hybrid. Hurricane withholds CSV history for an unbound dependency (T2), incompatible applicability (T4), or failed public validation (T6), then writes successful solutions where Hybrid fails (Appendix L). Only T6 invokes execution validation: it detects a delimiter error, not the separate quoted-newline defect later found by private grading.

5 Discussion and Conclusion

Hurricane separates retrieval relevance from reuse eligibility: records expose evidence, and admission checks it against current state. The experiments reveal four trade-offs.

First, stricter admission is not uniformly beneficial. Full admission leads in the twelve-task cohort but trails Hybrid and no-validation in an earlier diagnostic (Appendix E). These ablations do not isolate individual checks: components interact, and some changes reroute admission without changing the initial input.

Second, traceability has costs. Source allocation reduces token savings, and admission consumes tool time; fewer model tokens need not reduce total system cost.

Third, the best storage representation need not be the best reader-facing format. With QA evidence fixed, structured records lead on HotpotQA while prose leads on 2Wiki; wording and token length may also contribute. The matched API log shares Hurricane's initial input, so outcome differences do not establish a format benefit.

Fourth, position effects depend on task: Prepend improves Parents, but BFS is mixed. These component findings motivate task-specific choices; they do not establish a general benefit in deployed workflows.

Limitations. The API cohorts are constructed and run once per task and condition; their counts describe these fixed tasks, do not estimate run-to-run variability, and do not establish expected success on natural workloads or quality non-inferiority. The twelve-task cohort is exploratory. The six-task top-1 token interval crosses zero, so no token advantage is established. QA lacks paired uncertainty; long-context reports five-run means without variability estimates. Missing usage limits cost coverage. Frozen archives leave online writeback and cumulative improvement unevaluated. Admission checks public contracts, not semantic correctness, and cannot certify hidden state or eliminate hallucinations. Future work should test repeated paired runs on natural tasks and evaluate online writeback.

Reproducibility Statement

Reproduction details appear in Section 3 (records), Appendix A (QA and long context), Appendices B, G, and G.1 (API studies), Appendix J (operational interfaces), and Appendix K (input audits). Each cohort identifies remaining configuration, usage, and uncertainty limitations.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023. URL https://arxiv.org/abs/2310.11511.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025. URL https://arxiv.org/abs/2504.19413.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 758–759, 2009. doi: 10.1145/1571941.1572114. URL https://doi.org/10.1145/1571941.1572114.

Google DeepMind. MRCR v2: Evaluation and scoring, 2026. URL https://github.com/google-deepmind/eval_hub/tree/master/eval_hub/mrcr_v2. Reference implementation; experimental revision recorded in the source manifest.

Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2007. URL https://www.cs.ucdavis.edu/~green/papers/pods07.pdf.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland. MemoryArena: Benchmarking agent memory in interdependent multi-session agentic tasks. *arXiv preprint arXiv:2602.16313*, 2026. URL https://arxiv.org/abs/2602.16313.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020. URL https://aclanthology.org/2020.coling-main.580/.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2404.06654.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024. doi: 10.18653/v1/2024.acl-long.91. URL https://aclanthology.org/2024.acl-long.91/.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25961–25970, 2025. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

Martin Leucker and Christian Schallhart. A brief account of runtime verification. *Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009. doi: 10.1016/j.jlap.2008.08.004. URL https://christian.schallhart.net/publications/2009--jlap--a-brief-account-of-runtime-verification.pdf.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.nips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Bertrand Meyer. Applying Design by Contract. *Computer*, 25(10):40–51, 1992. doi: 10.1109/2.161279. URL https://doi.org/10.1109/2.161279.

OpenAI. GraphWalks. Revised dataset release, pinned revision be6cc6ecf9b4d495b07d1ff53d2a16598e90fed7, 2026. URL https://huggingface.co/datasets/openai/graphwalks.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=GN921JHCRw.

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

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2309.02427.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, 2026. doi: 10.18653/v1/2026.eacl-long.333. URL https://aclanthology.org/2026.eacl-long.333/.

Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, Rakesh Shivanna, Jeffrey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, Rohan Anil, Ethan Dyer, Siamak Shakeri, Roopali Vij, Harsh Mehta, Vinay Ramasesh, Quoc Le, Ed Chi, Yifeng Lu, Orhan Firat, Angeliki Lazaridou, Jean-Baptiste Lespiau, Nithya Attaluri, and Kate Olszewska. Michelangelo: Long context evaluations beyond haystacks via latent structure queries. *arXiv preprint arXiv:2409.12640*, 2024. URL https://arxiv.org/abs/2409.12640v2.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

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 of *Proceedings of Machine Learning Research*, pp. 63897–63911, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mlJLVigNHp.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024. URL https://arxiv.org/abs/2401.15884.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://hotpotqa.github.io/.

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. URL https://arxiv.org/abs/2210.03629.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, 2026. doi: 10.18653/v1/2026.acl-long.981. URL https://aclanthology.org/2026.acl-long.981/.

Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng Yan. G-Memory: Tracing hierarchical memory for multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/136a45cd9b841bf785625709a19c6508-Abstract-Conference.html.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. MemSkill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026a. URL https://arxiv.org/abs/2602.02474.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=eC4ygDs02R.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. MemRL: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026c. URL https://arxiv.org/abs/2601.03192.

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.

Weixiang Zhao, Yingshuo Wang, Yichen Zhang, Yang Deng, Yanyan Zhao, Wanxiang Che, Bing Qin, and Ting Liu. Large language model agents are not always faithful self-evolvers. *arXiv preprint arXiv:2601.22436*, 2026. URL https://arxiv.org/abs/2601.22436.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946. URL https://ojs.aaai.org/index.php/AAAI/article/view/29946.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning LLM agents without fine-tuning LLMs. *arXiv preprint arXiv:2508.16153*, 2025. URL https://arxiv.org/abs/2508.16153.

Xu Zou and Jie Tang. Trace as state: Reasoning traces as conditional states for long-context transformers. *arXiv preprint arXiv:2609.02702*, 2026. URL https://arxiv.org/abs/2609.02702.

A QA Organization and Same-Task State Protocols

The complete QA and long-context experiment protocols appear first. Historical partial-run and API diagnostic cohorts elsewhere in the appendices are reported separately and are not pooled with the main results.

Models, providers, and defaults. DeepSeek-V3 is the predominant backend. Studies without another model label use SiliconFlow's `deepseek-ai/DeepSeek-V3` endpoint. Labeled DeepSeek-V4-Flash runs use `deepseek-ai/DeepSeek-V4-Flash`, also served by SiliconFlow. Qwen models use Alibaba Cloud Bailian/DashScope; GLM models use the official GLM API. Provider defaults apply unless a study reports an explicit setting. Such settings and per-task execution budgets are documented with each cohort.

A.1 Complete QA organization experiment

Representations and matching. The five conditions in Table 1 use the same target questions and source executions. Raw preserves observable source logs; Summary and Structured use separate model calls to organize those logs. The organizer receives source actions and public observations, excluding hidden reasoning, submitted answers, and grades. Each target is executed once per condition.

Same-evidence prose is a deterministic rendering of the Structured record without another model call. It preserves fact text, evidence identifiers, categories, entry order, and source identifiers. All conditions append history under a 4,096-token cap. If either of the two content-matched representations exceeds the cap, the same entries are removed from both until they fit. This procedure controls retained evidence, not token length. The independently generated Summary is a separate organization condition.

Execution and budget control. The QA executor uses SWE-Agent and DeepSeek-V3. Ordinary target limits are eight model calls, 160,000 tokens, and 600 tool seconds. Source preparation and the organization required by a condition are allocated over three targets in a source group. Raw + budget reallocates the organizer allowance to the first target in each group: nine calls, 200,000 tokens, and 660 tool seconds; the other two targets retain the ordinary limits. This control matches the per-group allowance rather than realized computation. Same-evidence prose shares Structured's organization and its allocated cost.

Scoring and interpretation. The unmodified official HotpotQA and 2Wiki benchmark scorers evaluate the submitted answers. EM measures exact-match answers and F1 averages target-level token F1; failed or unsubmitted targets remain in scoring. Table 1 reports the final condition-level percentages. The organization comparison can change both selected content and its form; the deterministic prose control holds the former fixed.

A.2 Completed multi-model long-context experiment

Models, conditions, and repeats. The completed experiment evaluates DeepSeek-V4-Flash, Qwen 3.7 Max, and GLM-5.2 on BFS, Parents, and MRCR v2. Each model has three conditions: First pass without injected history, Hurricane Prepend, and Hurricane Append. Each reported score is the mean percentage over five repeats of the full pipeline. For each question, each repeat reruns First pass and the organizer, then evaluates both position branches using that repeat's shared state T. All evaluated samples contribute to the results; requests that time out are retried. Thus Table 2 is not conditioned on a subset of returned responses or on answer correctness.

Tasks and context sizes. The GraphWalks-256K cohort comprises 200 BFS and 200 Parents questions at the reported 256K-token context setting. MRCR v2 uses the eight-needle setting, with 200 questions at 256K and 200 at 512K. Its single table column pools these 400 question–length cases within each repeat, then averages over five repeats. Length-specific scores are not reported. These are nominal task-context settings, not measured complete-request lengths or model window capacities.

API configuration. DeepSeek-V4-Flash uses SiliconFlow; its endpoint is specified above. Qwen 3.7 Max uses official Alibaba Cloud Bailian/DashScope; GLM-5.2 uses the official GLM API. Each model performs its own First pass, organization, and subsequent solving; the organizer is not shared across models. DeepSeek-V4-Flash enables thinking for First pass, Prepend, and Append, and disables it for organization. The runs use post-August 2026 model versions. Qwen and GLM use their providers' default settings at all stages. Defaults need not coincide across services. Exact model revisions and resolved default parameter values are unspecified.

State and position intervention. Within each repeat, the first execution supplies same-task state for the organizer. The newly constructed T is shared by both position branches, with the original input x and question q retained: $[T, x, q]$ versus $[x, T, q]$. The position comparison holds the state content fixed. The model weights remain unchanged, and neither second-pass branch writes into the other's history. Comparisons with First pass also change computation: each second-pass condition uses a first execution, organization, and another solve. Prepend–Append comparisons vary state placement. Conditions use aligned data and task instances, with the same state in both position branches. When a historical reasoning trace exceeds 50,000 characters, only its first 50,000 characters are retained, measured directly by character count without tokenization. This cap applies to the trace, not the original task context or the complete request. Actual full-input lengths and truncation frequencies are not reported here. This experiment isolates placement of organized state; it does not include a raw-trace representation control.

Metrics and aggregation. Graph tasks use the unmodified official GraphWalks scorer for node-set exact match and F1. Both the complete and local MRCR experiments use the archived official `mrcr_v2_metric` function for Seq. We additionally report EM, comparing the complete visible response with the complete reference, removing only leading and trailing whitespace from each. The required marker is retained; no answer is extracted. For Seq., the marker is the trimmed reference's first 12 characters. The scorer extracts the text after its last occurrence in the response and removes the marker from the reference. Both strings are stripped of leading and trailing whitespace before SequenceMatcher compares the reference as its first argument with the response as its second. A missing marker scores zero. Thus EM and Seq. evaluate different text spans and are not interchangeable. Higher scores are better. Table 2 bolds the maximum reported score for each model–metric column, including ties. The figure and text calculate percentage-point differences from the displayed means. We report five-repeat means and descriptive differences without repeat-level variability estimates, confidence intervals, or significance tests. The earlier partial run below uses a separate cohort and protocol; its sample counts, additional serialization limits, and decoding budgets do not apply here.

A.3 Earlier partial-run protocol and availability audit

This separate exploratory study uses a smaller cohort and different context lengths. Its observations and denominators are not pooled with the completed multi-model results in Section 4.3.

Earlier partial-run source and serialization. The earlier partial DeepSeek-V4-Flash study uses a frozen set of thirty questions from each of BFS, Parents, and MRCR v2, drawn from the existing 128K-and-shorter cohort. It is an exposed exploratory subset, not a newly preregistered independent sample. First, Prepend, and Append request thinking mode, maximum reasoning effort, a 16,384-token thinking budget, a 4,096-token answer limit, and temperature zero. These are requested API settings, not guarantees of deterministic output. The organizer uses the same model without thinking, JSON output, and a 4,096-token limit. No tool executor is used in this cohort.

Only the first response's provider-returned reasoning field enters the organizer. Its first 50,000 characters are divided into 2,000-character events. Each admitted state entry has nonempty text, known event identifiers, and a quotation of at least ten characters appearing in a cited event. At most four entries per field are admitted. Missing permitted lists are normalized to empty lists; unknown fields are rejected. At least one entry must survive. An organized JSON state exceeding 50,000 characters is rejected rather than cut mid-record. These limits apply to the trace and state, not to the original task context. The state is tagged same-task-only and unverified, wrapped with the trace markers and warning text adapted from Zou & Tang (2026), and reused byte-for-byte in the two position arms. The complete original context and final question are identical across arms.

Earlier partial-run scoring and availability. Graph scoring uses the frozen answer parser and node-set EM/F1. MRCR uses the same full-response EM and marker-based Seq. rules as the completed study (Appendix A.2). Seq. accepts a marker occurring later in the response, whereas EM requires the complete stripped response to match. Scoring requires labels, but neither source organization nor state construction receives them. A returned response is distinct from a nonempty answer or valid submission. In the 222 returned answer cells, one first-pass answer is empty and three responses terminate at the output limit. All are retained under the frozen scorer.

All ninety question pipelines have terminal states. Seventy-six first passes and seventy-six organizers return; seventy-five Prepend and seventy-one Append results return. The 383 physical calls include 298 successful deliveries and 85 failed deliveries without final usage. There are 42 unattempted logical phases after upstream failure. Transport recovery permits at most three physical attempts per logical request, independent of answer quality. No retry is triggered by an incorrect answer or a formatting failure. Protocol amendments normalize empty fields and recover transport; they do not choose prompts or state by correctness. Sequential branch execution and provider/time drift remain potential confounds, and missing responses need not be random. Known usage is 10,954,725 input and 1,724,225 output tokens; partial-stream counters are lower bounds, not final invoices. No exact cost comparison is made from these incomplete records.

Historical response-level analysis. The historical analysis script checks the 270 scheduled long-context rows, derives the tables, and records hashes of source reports. It retains the 71 common position pairs and the full ninety-question denominator as separate estimands. An additional response-level audit checks output hashes, state hashes, and identical input blocks. Frozen experiment files are not edited to prepare the manuscript. Earlier cross-task graph results use another acquisition procedure and decoding configuration and are not combined with the same-task results.

B API Policy Comparison: Protocol and Accounting

The archived API reports describe separate cohorts and retain all scheduled slots, failed calls, source records, and initial-message audits. The API cohorts use constructed development tasks. Table 7 breaks down Table 3 by family to show its clustered design. The frozen exploratory plan designated Hurricane versus Self-Refine as the primary comparison, with Hybrid and the matched log as strong controls. The exposure, success, and token comparisons in Section 4.4 are descriptive analyses; the frozen plan did not specify a non-inferiority or efficiency test.

Task construction and historical access. Table 3 crosses six API families with six reuse scenarios; the 36 targets are related development cases rather than 36 independent source lineages. Each execution starts with an unimplemented `solve(value)`, a public contract, options, and visible examples. The historical archive contains one source implementation per family and is frozen before these target runs. Source answers are not target labels. The actor can inspect files, write or adapt code, run public examples, and submit; private probes are used only by the evaluator.

The stable scenario retains the original conditions. Missing-dependency cases omit the recorded `options.json` dependency; transitive-change and applicability cases change the required mode. Expired-receipt cases apply an immediate-expiry policy to genuine source validation, while coverage cases add a public example beyond the source receipt's checks. These scenarios test admission and adaptation; their public inputs need not all differ, and the actor does not receive the scenario label. Revalidation executes historical code against public examples and restores the target stub before the actor begins. Its traces are kept in a separate audit, not supplied as initial tool feedback.

The access audit records a nonempty historical pack for 36/36 Hybrid targets, 15/36 Hurricane targets, 15/36 matched-log targets, and none of the no-history or Self-Refine targets. Hurricane and the matched log have identical initial inputs on every target. Self-Refine and no history also start with identical inputs; their intervention occurs after a draft write. These exposure counts measure available history, not proven use of it.

Self-Refine implementation. The baseline is adapted from the Self-Refine method (Madaan et al., 2023) to the same native SWE-Agent task loop. An eligible successful draft write can trigger feedback while at most five model calls have been used and fewer than two feedback rounds have occurred.

Condition	BFS EM / F1	Parents EM / F1	MRCR EM / Seq.
All planned questions: $n = 30$ per task; missing responses score zero			
First pass	66.7 / 66.7	66.7 / 76.6	56.7 / 63.9
Hurricane Prepend	73.3 / 73.3	83.3 / 83.3	66.7 / 73.2
Hurricane Append	76.7 / 76.7	60.0 / 67.7	60.0 / 60.5
Jointly returned position pairs: $n = 24$ BFS, 23 Parents, 24 MRCR			
First pass	83.3 / 83.3	78.3 / 91.3	62.5 / 71.5
Hurricane Prepend	91.7 / 91.7	100.0 / 100.0	75.0 / 83.1
Hurricane Append	95.8 / 95.8	78.3 / 88.4	75.0 / 75.6

Table 5: Earlier partial-run organized same-task state with DeepSeek-V4-Flash, inputs from the 128K-and-shorter cohort. Scores are percentages. The first block measures scheduled execution outcomes, including transport failures; the second uses the same completion-selected questions for all three rows. Both position arms share one state. First pass uses less computation. No raw same-task-trace baseline is included.

Task	First returned	Prepend returned	Append returned	Joint pairs
BFS	25/30	24/30	24/30	24/30
Parents	25/30	25/30	23/30	23/30
MRCR v2	26/30	26/30	24/30	24/30

Table 6: Delivery coverage. Returned means an API result exists, not that the answer is correct, nonempty, or format-valid. All 90 scheduled questions reached terminal states; this is not complete delivery of all 270 answer slots.

Feedback sees the current public task and draft only, requests a concise JSON response with a feedback string and stop boolean, and explicitly forbids invented test outcomes. It is capped at 180 words. A stop judgment can halt further feedback but is not a submission or success grade. Feedback, revision, and submission share the eight-call cap. The observed success rate applies to this adaptation and budget; the original Self-Refine study used different tasks and execution conditions.

Infrastructure and accounting. The 36-target policy comparison preserves one interrupted Self-Refine datetime target. All originally completed records were retained on resumption. A common-target sensitivity excludes that slot in every arm, rather than excluding only a baseline failure. This gives 31/35 for Hurricane and Hybrid, 29/35 for the matched log and no history, and 24/35 for Self-Refine. The archive records an unresolved model usage entry and unfinished tool-accounting operations from the incident. Therefore the campaign has valid quality scoring and execution-fairness checks but not a complete exact-cost ledger. Target-only known-token totals are 233,378 for Self-Refine, 228,481 for no history, 451,520 for Hybrid, 162,539 for Hurricane, and 183,914 for the matched log. The known target-execution subtotal is 1,259,832 model tokens across 756 recorded physical attempts, plus one invocation with unresolved usage.

Source-inclusive model-token comparison. Each memory condition is allocated the same 42,118 source-acquisition tokens once, giving 493,638 for Hybrid, 204,657 for Hurricane, and 226,032 for the matched log. No-history and Self-Refine require no source acquisition, so their known totals remain 228,481 and 233,378. Source acquisition was executed once physically; allocating it to each hypothetical strategy does not add another charge to the campaign total. The audit records zero unresolved target usage calls for Hybrid, Hurricane, the matched log, and no history, and one for Self-Refine, marked by † in Table 3. The known-token reductions for Hurricane relative to Hybrid are $1 - 162{,}539/451{,}520 = 64.0\%$ for targets and $1 - 204{,}657/493{,}638 = 58.5\%$ including sources, rounded to one decimal. These are measured model-token differences in a single-run cohort; they do not include tool, infrastructure, or monetary costs, and equal observed success does not establish statistical non-inferiority.

Wrong-transfer uncertainty. All five arms have zero confirmed cases under the strict historical-code proxy. Uncertain failures number 12 for Self-Refine, 6 for no history, 4 each for Hybrid and Hurricane, and 6 for the matched log. The same failure flag appears in no-history conditions, so it

Family	Self-Refine	No history	Hybrid	Hurricane	Matched log
decimal	5/6	6/6	6/6	6/6	6/6
datetime	5/6	6/6	6/6	6/6	6/6
csv	1/6	2/6	3/6	4/6	2/6
url	5/6	6/6	6/6	6/6	6/6
json	4/6	4/6	5/6	4/6	4/6
IP networks	4/6	6/6	6/6	6/6	6/6

Table 7: Self-Refine comparison by constructed family; six targets per family. These are clustered observations, not six independent draws within each row.

cannot identify adoption of historical content. These flags are therefore reported as uncertainty rather than comparative harmful-transfer rates. Attribution would require a counterfactual comparison that isolates the effect of historical content.

C Public-Task Presentation and Trace Studies

These archived studies test distinct presentations and workloads. HotpotQA (Yang et al., 2018) and 2WikiMultihopQA (Ho et al., 2020) use provided distractor contexts in the archived public-memory experiment, not open retrieval over all Wikipedia. AIME2024 is scored by exact integer answer. Source memories are frozen before targets. Names "Trace" and "Compact" were adopted after development inspection and map to original identifiers `no_compact` and `spega`; no data were rerun under the renamed labels. The Trace/Compact comparison changes one field of an otherwise fixed record: Compact omits optional observation detail τ, potentially allowing more records to fit within the history budget. This field-omission comparison is distinct from the independently generated prose and structured records in the QA organization study.

Trace selects two records on HotpotQA and one–two on 2Wiki, compared with four–five and two–five for Compact. Thus field omission changes both content detail and record coverage. Source-inclusive model tokens for Hybrid/Trace/Compact are 58,170/44,086/52,805 on HotpotQA, 106,795/92,077/97,987 on 2Wiki, and 184,820/206,165/184,671 on AIME. Trace therefore has lower observed model-token use at the same QA success counts, but higher use with higher AIME accuracy. These are neither powered non-inferiority results nor exact all-tool cost comparisons.

Table 11 reports the separate new-question QA placement study.

The native study exposes model-generated reasoning fields, whereas the API state study extracts observable tool state. They are distinct trace formats. Graph tasks benefit over the fresh no-trace repeat in this small cohort; MRCR shows no added EM. These 32K–64K observations are separate from the completed multi-model experiment in Section 4.3.

Separate API and public-QA placement studies. The API study shares the same AI-generated memo within each position pair, but direct conditions omit the first-pass/recorder workflow. Every first execution yields a nonempty memo; historical cross-task material is injected on 15/36 targets. First execution and recording are included in chain usage. The new-question QA study uses eight questions per dataset, excluding earlier question IDs and context titles. It is a small development component study, separate from the complete QA organization comparison in Section 4.2.

D Fixed-Routing Memory Competition

Twelve exposed synthetic fixtures share the candidate archive, automatic router, SWE-Agent, DeepSeek-V3, and per-task limits. Every method solves 12/12. Model-token totals are 310,164 (Flat), 312,422 (BM25), 316,765 (Dense), 309,689 (Hybrid), 288,608 (matched log), 280,211 (Hurricane-Compact), and 320,619 (Compact without links). An empty-pack control also solves 12/12 using 271,226 tokens, so this ceiling cohort does not establish a benefit from supplying memory. Compact is 9.5% below Hybrid and 2.9% below the matched log in this token measure. Exact I/O totals are incomplete, and equal observed success on twelve ceiling fixtures does not establish quality

Method	HotpotQA (8)	2Wiki (8)	AIME24 (30)
Vanilla reflection	25.0	37.5	30.0
Strong single-store Hybrid	50.0	50.0	20.0
Ordinary log + matched policy (Compact)	25.0	50.0	23.3
Hurricane-Trace	50.0	50.0	30.0
Hurricane-Compact	25.0	50.0	6.7
Compact without links	25.0	37.5	26.7
Compact without templates	12.5	62.5	23.3
Compact without loop	37.5	25.0	0.0

Table 8: Archived public-memory development results: EM or integer accuracy in percent. The underlying archive contains 414 executions across nine conditions; the eight memory-bearing configurations are shown here. Ablations refer to Compact, not Trace. Removing the loop also changes generation opportunities.

Task	First	No-trace repeat	Prepend	Append
BFS	55.0 / 57.5	37.5 / 44.5	75.0 / 75.0	62.5 / 62.5
Parents	57.5 / 68.7	75.0 / 95.1	87.5 / 87.5	100.0 / 100.0
MRCR	77.5 / 97.6	87.5 / 100.0	87.5 / 100.0	87.5 / 100.0

Table 9: Separate native-trace component study using DeepSeek-V4-Flash and complete 32K–64K-token inputs, eight questions per task. Values are percentages: EM / set F1 for graph tasks and EM / sequence similarity for MRCR. First averages five executions within each question; subsequent arms receive one call and share that source cost. Rounded sequence similarity is not exact match.

non-inferiority. This diagnostic is retained as a fixed-routing cost observation, not evidence for independent real-source quality or exclusive dual-memory capability.

E Earlier Mechanism Diagnostic

An earlier, separately frozen diagnostic runs eight arms on the same family design, for 288 target executions. Input-level audits identify which nominal ablations change delivered information, keeping this cohort separate from the 36-target policy comparison in Table 3.

The full gate solves 29/36, versus Hybrid's 35/36. Removing validation solves 33/36. Thus this cohort does not establish that stricter admission improves task success. A rejected historical artifact can still contain useful clues, and the remaining public task may not let the model reconstruct them within the cap. That is a possible explanation, not a measured causal decomposition.

Removing dependency closure changes routes on six targets and initial inputs on five; full admission and no closure both solve those five. Removing applicability changes thirty routes but no initial inputs, because other checks still determine admission. Removing validation changes inputs only on CSV/T1, CSV/T5, and CSV/T6: full admission solves 1/3, compared with 3/3 without validation. Of the four additional successes without validation overall, two occur among these changed-input targets and two among the 33 unchanged-input targets. Thus the aggregate difference does not isolate the benefit of supplying previously rejected history. Hurricane and the matched log have identical initial inputs on all 36 targets and equal aggregate success, yet differ on four paired outcomes. These post-hoc comparisons distinguish changes in delivered information from variation between single executions; route counts alone cannot do so.

F Matched-Source Acquisition Audit

Cohort and exposure. The 100 BFS targets are selected from the official pinned `graphwalks_128k_and_shorter.parquet` file by a fixed salted hash of the public-input digest and row index. The cohort was reduced from an earlier, larger evaluation queue after pilot and interim outcomes had been inspected. Although selection does not use those outcomes, this remains an exploratory cohort rather than an untouched confirmatory sample. The source is a separate task

Condition	Success	Rate (%)
Hurricane + state prepend	27/36	75.0
Hurricane + state append	28/36	77.8
Hurricane direct	31/36	86.1
Hybrid direct	28/36	77.8

Table 10: Separate API state-placement cohort. The same AI-generated memo is used in each prepend–append pair. Direct conditions have the same overall task cap; Appendix L analyzes cases from these same direct runs. These are not long-context reproduction results.

	HotpotQA ($n = 8$)			2Wiki ($n = 8$)		
Condition	EM	F1	Tokens	EM	F1	Tokens
First	25.0	33.3	107,916	37.5	45.8	114,967
Direct	62.5	76.4	115,374	62.5	62.5	110,635
Strong Hybrid	62.5	70.8	98,313	50.0	58.3	118,258
State append	87.5	95.8	147,733	62.5	66.7	167,621
State prepend	62.5	77.1	159,391	75.0	75.0	175,467

Table 11: New-question QA results with SWE-Agent and DeepSeek-V3. EM and F1 are percentages; tokens are source-inclusive chain totals for each dataset. First alone has a shorter budget. Neither placement was chosen prospectively.

Condition	Success	Rate (%)
No history / no extra feedback	28/36	77.8
Hybrid	35/36	97.2
Ordinary log + hash check	30/36	83.3
Hurricane	29/36	80.6
Ordinary log + same checks	29/36	80.6
Hurricane without closure	29/36	80.6
Hurricane without applicability	30/36	83.3
Hurricane without validation	33/36	91.7

Table 12: Earlier mechanism diagnostic, separately frozen and executed. Every scheduled slot is retained. These ablations do not show a uniform gain from stricter admission.

Experience	EM (%)	F1 (%)	Output-limit hits
None	44.0	48.17	20
Raw source response	46.0	50.81	13
Prose summary	45.0	49.85	21
Structured experience	43.0	46.11	22

Table 13: Archived BFS acquisition diagnostic, 100 targets per condition. The shared source contains only an empty-list final answer. These results test a deficient acquisition configuration, not substantive procedure distillation. They are separate from the same-task state study.

and is shared across all experience conditions. There is one source realization, not 100 independent experience sources. The model is `deepseek-ai/DeepSeek-V4-Flash`, temperature zero, with provider thinking disabled; those settings do not guarantee identical responses on repeated executions.

Scoring and failures. The frozen parser requires a final line containing `Final Answer:` and a bracketed node list. Exact match compares node sets, and F1 is the harmonic mean of node precision and recall. Empty prediction and empty reference receive one; a missing or invalid answer receives zero. Of 100 targets, 17 have an empty reference set. No experience, raw response, prose summary, and structured experience solve 12, 13, 13, and 10 of these, respectively. On the remaining 83 targets they solve 32, 33, 32, and 33. All 400 API requests return normally, although 20, 13, 21, and 22

responses reach the output limit. Format-invalid counts are 23, 14, 21, and 22. The primary scores retain all failures; no alternative parser is substituted. Structured experience versus raw response has 10 wins, 13 losses, and 77 ties; versus prose summary it has 14 wins, 16 losses, and 70 ties.

Revised acquisition contract. A subsequent implementation removes answer-only instructions from the source acquisition request. It requests two to six distinct observable quotations, two to five reusable procedural statements with observation references, and an explicit list of unresolved checks. Admission verifies that quotations occur verbatim in the source context, observation IDs are unique, procedure entries are nonempty, and cited IDs exist. The structured organizer must retain observations and procedure; unsupported applicability or dependency fields may remain empty. A pack that would require truncation is rejected under this revision. These are evidence-integrity checks, not a semantic proof of the procedure or a claim that model narration is tool telemetry. Target labels and target outcomes are unavailable to admission.

Scope of the revised contract. The implementation passed six model-free tests covering answer-only records, missing procedures, fabricated quotations, invalid references, valid records, and prompt-contract consistency. A subsequent source request produced incomplete JSON and was rejected before target execution. No downstream performance is therefore reported for this revision. The results in Table 13 describe the original acquisition configuration.

G Exploratory API History-Cap Sweep

Cohort and selection. The initial six-task efficiency study uses SWE-Agent 1.1.0 and DeepSeek-V3 with the same eight-call, 160,000-token, 600-tool-second target limits across conditions. The execution budget was reduced to 100 target executions; four already-dispatched executions also completed. Selection follows reservation timestamps, with the amendment recorded before private grading. Of the first 100 executions, 96 form six complete task blocks, each with five memory methods at three history caps plus one no-history condition. These are six paired targets, not 96 independent tasks or three repeats per condition. The targets span two development families (decimal and CSV) and four new constructed families (decompression, Base64 decoding, HTML escaping, and Unicode normalization), with one scenario instance per family. The later extension adds a second instance from each of these six families (Appendix G.1); it is not an independent replication.

Observed comparisons. At the 4,096-token cap, Hurricane and Full Hybrid both solve 6/6 targets, using 3,363 and 13,306 known target tokens per target, respectively (rounded means). The difference before rounding is 9,942.5 tokens per target, and target usage is complete for both configurations. Hybrid top-1, the matched log, and the no-validation condition also solve 6/6 at this cap. Hybrid top-1 uses 3,751 target tokens per target. Hurricane uses 388 fewer target tokens per target (10.3%). The six-task 95% paired bootstrap confidence interval for Hurricane minus top-1 is $[-1{,}073.5,\ 250.0]$ tokens, including zero. After shared acquisition is added, the reduction is 1.26% of the known source-inclusive subtotal. No history solves 5/6 and incurs no source acquisition. Hurricane's target reduction relative to Full Hybrid is 74.7%. The known source-inclusive means are 30,418 for Hurricane, 40,361 for Full Hybrid, 30,806 for top-1, 31,330 for the matched log, and 30,617 without validation. Known usage from failed calls is retained; unknown usage is not assigned zero. These observations do not establish dominance over simpler history policies or non-inferiority.

Interpreting the top-1 comparison. The point estimate favors Hurricane, but the interval permits both token savings and token increases relative to top-1. It neither resolves the direction of the cost difference nor establishes equivalence. All six tasks succeed under both methods, so this cohort provides no observed success-rate separation. We therefore treat top-1 as a competitive compact-history baseline. The twelve-task extension below tests success under a broader set of instances; its success difference does not establish a token-cost advantage on those twelve tasks.

Effective inputs. Packing admits only whole records. At cap 256, no memory condition fits a record on any of the six targets; each has the same initial messages as no history. For Hurricane, Hybrid top-1, the matched log, and no validation, initial messages are also identical between caps 1,024 and 4,096 on every target. These matches concern initial messages, not subsequent tool calls or

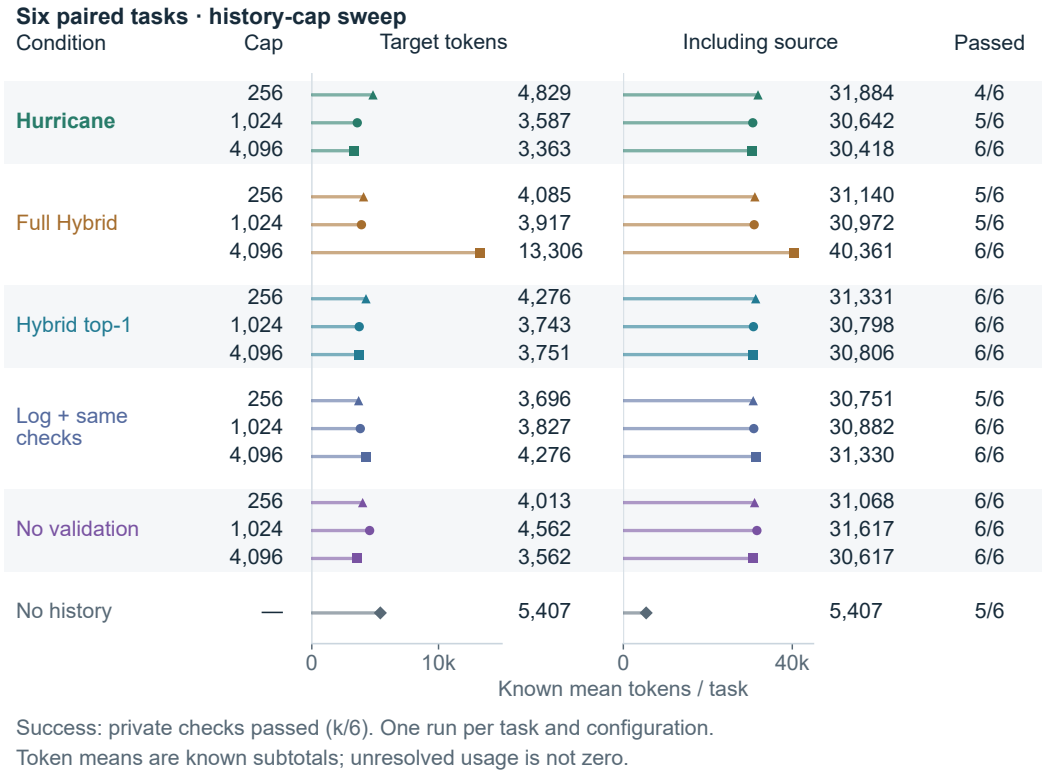


Figure 4: History-cap sweep on the original six paired API tasks: sixteen configurations and 96 executions, with one execution per task and configuration. Rows show the cap, known mean target and source-inclusive tokens, and private-check success ($k/6$). The cost columns use separate zero-based axes; printed means are rounded to integers. No history is the common reference condition. Known failed-call usage is retained; unresolved usage is not treated as zero.

returns. Differences across these settings therefore do not measure an effect of additional history in the initial context. Table 14 reports nonempty initial packs and configured caps.

Condition	Cap	Nonempty initial pack	Success	Target tokens	Including source tokens	Unknown target calls
Hurricane	256	0/6	4/6	4,829	31,884	1
Hurricane	1,024	2/6	5/6	3,587	30,642	1
Hurricane	4,096	2/6	6/6	3,363	30,418	0
Full Hybrid	256	0/6	5/6	4,085	31,140	0
Full Hybrid	1,024	6/6	5/6	3,917	30,972	1
Full Hybrid	4,096	6/6	6/6	13,306	40,361	0
Hybrid top-1	256	0/6	6/6	4,276	31,331	0
Hybrid top-1	1,024	6/6	6/6	3,743	30,798	0
Hybrid top-1	4,096	6/6	6/6	3,751	30,806	0
Log + same checks	256	0/6	5/6	3,696	30,751	1
Log + same checks	1,024	2/6	6/6	3,827	30,882	0
Log + same checks	4,096	2/6	6/6	4,276	31,330	0
No validation	256	0/6	6/6	4,013	31,068	0
No validation	1,024	3/6	6/6	4,562	31,617	0
No validation	4,096	3/6	6/6	3,562	30,617	0
No history	–	0/6	5/6	5,407	5,407	0

Table 14: Complete-block subset of the stopped history-cap sweep: sixteen configurations on six paired tasks (96 executions). Cap denotes the initial history budget in tokens; pack counts do not describe later history-tool access. Token columns are known mean tokens per target, rounded to the nearest integer. Unknown usage does not remove an execution from success scoring. All memory configurations also share four source calls with unresolved token usage; no-history incurs no source acquisition.

Top-1 comparison by delivered history. A post-hoc input audit separates all six paired targets without selecting on outcomes. At caps 1,024 and 4,096, Hurricane and top-1 have byte-identical initial messages on CSV/T1 and Unicode/T5, with matching recorded history-tool IDs. On decompression/T4, Base64/T3, decimal/T3, and HTML/T1, Hurricane supplies no history while top-1 supplies its leading record. At cap 4,096, both policies solve these four targets. Their mean target costs are 3,666.25 and 4,255.75 tokens, respectively: 589.5 fewer tokens for Hurricane (13.9%), with complete target usage. At cap 1,024, the same four targets yield 3/4 versus 4/4 successes; Hurricane also has one unresolved-usage call. Within each policy, initial messages and recorded history-tool IDs match across these two caps. The two executions therefore do not isolate an effect of added initial history or establish a stable success–cost advantage. This subset is exploratory and does not supply top-1 outcomes for the main API cohort.

Accounting and uncertainty. The source-inclusive view allocates all 36 completed source executions and archive preparation: 162,329 known tokens, or 27,054.8 per target in each memory configuration. It includes acquisition left unused by the early stop. Restricting the allocation to the two replicas actually used by the paired targets gives 107,164 known source tokens, or 17,860.7 per target; this sensitivity reduces every memory point by 9,194.2 tokens. Neither allocation is a general deployment amortization rate. Four source calls have unresolved token usage. Shared acquisition cancels in absolute differences between memory configurations, but source-inclusive percentages refer only to known subtotals. Physical shared acquisition is counted once, and the per-strategy allocations must not be summed as an invoice.

The cost analysis uses 10,000 paired task-block bootstrap resamples, conditional on the saved acquisitions. Resampling six observed successes yields a degenerate bootstrap success-rate interval; this does not establish population-level certainty. We report success counts descriptively.

Executions outside the paired display. Four other primary executions fail: no validation on Base64/T4 and decimal/T5, Hurricane on Base64/T4, and the matched log on decimal/T5. Each has one unknown-usage call; zero known tokens do not mean zero cost. The four overflow executions have three successes, 9,757 known target tokens, and one unknown-usage call. All eight records remain in the report. Complete-block selection can depend on execution time despite not using scores; the figure does not represent all 100 primary executions. Sources, preparation, and all 104 target

executions use 617,291 known tokens with thirteen unknown-usage calls. This cohort is not pooled with Table 3.

G.1 Expansion to twelve tasks: mechanism comparison

Cohort and stopping. The run was stopped on request. The expanded analysis includes twelve complete paired task blocks available after stopping, with two target instances from each of six families (Table 15). A complete block contains all sixteen method–cap settings for one target, including the shared no-history condition. CSV and Decimal are development families; HTML escaping, Base64 decoding, decompression, and Unicode normalization are additional constructed families. Inclusion depends on completion of the condition block, rather than task success. The cohort was not a prespecified confirmation set. Because block completion can depend on execution time, these results characterize the completed cohort rather than a random sample of tasks. The four additional families broaden coverage beyond CSV and Decimal; the extension adds a second instance within each of the same six families. This constructed evaluation does not establish generalization to natural workloads.

Family	Target behavior	Reuse scenario
CSV	Parse comma-separated text, including quotes and newlines	Unchanged configuration
HTML escaping	Escape special characters and quotation marks	Unchanged configuration
Decimal	Change rounding from half-up to half-even	Dependency configuration change
Base64	Change from strict to permissive decoding	Dependency configuration change
Decompression	Change the format from gzip to zlib	Applicability change
Unicode	NFC normalization; recheck an expired receipt	Validation receipt expiry

Table 15: Task families and reuse scenarios in the twelve-task extension, with two target instances per family. CSV and Decimal are development families; the other four are additional constructed families. Scenario labels describe the intended change, not observed gate decisions or outcomes.

Shared configuration and execution units. All conditions use SWE-Agent 1.1.0, DeepSeek-V3, and temperature zero, with the same candidate archive and ranking. Each target execution is limited to eight model calls, 160,000 tokens, and 600 tool seconds. Programmatic admission checks run before the first target model call and consume this tool allowance without making model calls. Hidden grading is separate, is not returned to the agent, and does not consume the allowance (Appendix J.2). Shared target budgets include admission checks; the tool time remaining for the reader can therefore differ across conditions. Hurricane, Full Hybrid, Hybrid top-1, the matched log, and Hurricane without validation are each evaluated at history caps of 256, 1,024, and 4,096 tokens; no history is evaluated once per target. This gives sixteen method–cap settings per target and $12 \times (5 \times 3 + 1) = 192$ target executions. The 192 executions do not constitute 192 independent tasks. Each task–condition–cap configuration runs once. The 96- and 192-execution counts describe the six- and twelve-task analysis matrices; they are not added to obtain a physical campaign total. Temperature zero does not establish deterministic execution.

Mechanism comparison. Table 4 compares three historical conditions at the 4,096-token cap with the shared no-history condition. In every condition and at every cap of this twelve-task matrix, the initial input and history tools use the same selected records; the remaining archive is inaccessible. Hybrid top-1 selects at most the highest-ranked record. Hurricane without validation retains the other admission rules and removes the validation check. Full Hurricane uses applicability, dependencies, and validation to determine which history is delivered. No history removes historical experience. Success requires the submitted result to pass the private checks, rather than matching a question-answering reference string.

Observed comparison. The task-level outcomes underlying Table 4 are Hurricane (full): 12/12 (100.0%); Hybrid top-1: 9/12 (75.0%); Hurricane without validation: 9/12 (75.0%); and No history: 5/12 (41.7%). The complete configuration therefore has three more successes (25 percentage points) than either top-1 or the validation ablation. The no-history comparison evaluates the reuse

configuration as a whole. The top-1 comparison asks whether selecting the leading record is sufficient; the validation ablation addresses the incremental role of that check within the remaining admission mechanism. The results do not separately identify the contributions of dependency and applicability checks. Equal aggregate scores for the two 9/12 conditions do not identify their failed tasks. The aggregate counts also do not locate the additional successes by task family.

Relationship to the efficiency study. The twelve-task evaluation extends the six-task cohort; it is not an independent replication or a twelve-family study. The six-task cost analysis and twelve-task mechanism summary concern related task sets but are reported separately. Outcomes on the six added tasks are not inferred by subtracting earlier counts. The twelve-task success table reports one cap and selected mechanism controls, rather than outcomes for all sixteen method–cap settings.

The twelve-task summary reports success outcomes only. The 74.7% reduction relative to Full Hybrid and the bootstrap interval for the top-1 token contrast describe the original six-task efficiency comparison, not all twelve tasks. No cost estimate, confidence interval, significance test, or claim of quality non-inferiority is derived from the twelve-task success table. The observed 12/12 versus 9/12 counts describe this completed, non-prespecified cohort only. With one run per task and condition, they do not establish that conditional admission generally improves on top-1 or establish a general quality–cost advantage over top-1.

H Representation–Position Diagnostic with Delivery Failures

We attempted a separate 3×2 diagnostic with DeepSeek-V4-Flash: raw trace, prose summary, and Hurricane state, each prepended or appended. BFS and Parents each have thirty planned questions, yielding sixty question blocks and 360 second-pass slots. A question shares its first-pass source across representations; each resulting state is frozen before the position comparison, under a 4,096-token memory cap. Each condition runs once per question. This batch is separate from the completed three-model, five-repeat study in Table 2.

Task	Representation	Position	EM (%)	F1 (%)	Reported returns	Known chain tokens / task
BFS	Raw trace	Pre	3.3	3.3	2/30	6,261
BFS	Raw trace	Post	3.3	3.3	1/30	3,422
BFS	Prose summary	Pre	6.7	6.7	2/30	6,733
BFS	Prose summary	Post	6.7	6.7	2/30	6,739
BFS	Hurricane	Pre	6.7	6.7	2/30	6,823
BFS	Hurricane	Post	3.3	3.3	2/30	7,086
Parents	Raw trace	Pre	16.7	16.7	5/30	17,816
Parents	Raw trace	Post	10.0	10.0	5/30	19,106
Parents	Prose summary	Pre	16.7	16.7	6/30	20,825
Parents	Prose summary	Post	13.3	16.2	5/30	20,968
Parents	Hurricane	Pre	13.3	16.7	6/30	18,069
Parents	Hurricane	Post	6.7	9.5	4/30	19,120

Table 16: Separate DeepSeek-V4-Flash representation–position diagnostic, with thirty planned questions per task and one run per question and condition. Pre and Post denote Prepend and Append. Scores, return counts, and costs reproduce the reported aggregate table; costs are known source-inclusive tokens per planned question. Failed and acquisition-blocked slots remain in the denominator. Differences from the receipt snapshot are discussed in Appendix H.

Delivery and scoring. All planned slots have a terminal disposition, but most do not reach a second-pass answer. The archived receipts show only 9/60 normal first-pass returns. Second-pass records contain forty normal returns, ten request failures, and 310 acquisition failures: 306 follow a failed first pass and four follow a failed organizer. Normal return denotes receipt status, not answer correctness. The archived scoring assigns zero to failures and retains all thirty planned questions per condition. Consequently, these scores describe the executed pipeline, including delivery failures, and cannot isolate representation quality. The low scores do not establish that any representation is intrinsically ineffective.

Result versions and accounting scope. Table 16 reproduces the final reported scores, return counts, and per-question known-token costs. For Parents/Hurricane/Pre, it reports 16.7% F1 and 6/30 normal returns. The preserved receipt snapshot instead records 13.3% F1 and 4/30 returns; EM is 13.3% in both versions. Reported returns sum to 42, whereas the failure categories above and physical-call totals below refer to the forty-return receipt snapshot. Updated trial records are not part of the release. The two versions remain separately identified; their aggregates are not combined to reconstruct paired outcomes or confidence intervals.

Cost and interpretation. The archive contains 128 physical requests, 2,348,551 known tokens, and 63 calls with unresolved usage. Shared first-pass and organizer costs are allocated to each logical condition, while physical calls are counted once. Missing usage is not zero cost, and acquisition-blocked slots incur no second-pass request. Thus differences in known tokens per planned question are not evidence of matched-quality efficiency. A reliable replication under a common, prospectively fixed delivery and retry policy is needed to resolve the representation comparison. The present batch is retained as a delivery diagnostic, with no superiority claim.

I HURRICANE SOFTWARE AND NAMING

Method and software names. *Hurricane* names both the framework and the current software version. The research studies experience acquisition, organization, and access conditioned on the current task, with dependency and validation checks as components of that process. Archived artifacts retain the identifier `SPEGA` and arm key `spega`; Hurricane denotes the same unchanged condition in the manuscript. Trace, Compact, and validation ablations retain their original meanings and study boundaries.

Product interface and evaluation boundary. Figures 5 and 6 show actual local Hurricane windows captured with the English locale on September 20, 2026. The application provides a user-facing shell for knowledge access, workflow configuration, and software-development tasks. These views document interface controls, while the controlled API studies evaluate SWE-Agent and DeepSeek-V3 under the frozen admission policy. They do not evaluate the complete desktop application or its other model configurations.

Knowledge access. The knowledge-library panel exposes import, export, synchronization, search, source inspection, and quality views. Its automatic-recall control and score threshold make the retrieval-to-context boundary visible to the user. The panel reports local SQLite storage and full-text search capabilities. This product view does not establish dual-vector-store selection or the benefit of the dependency checks studied in the experiments.

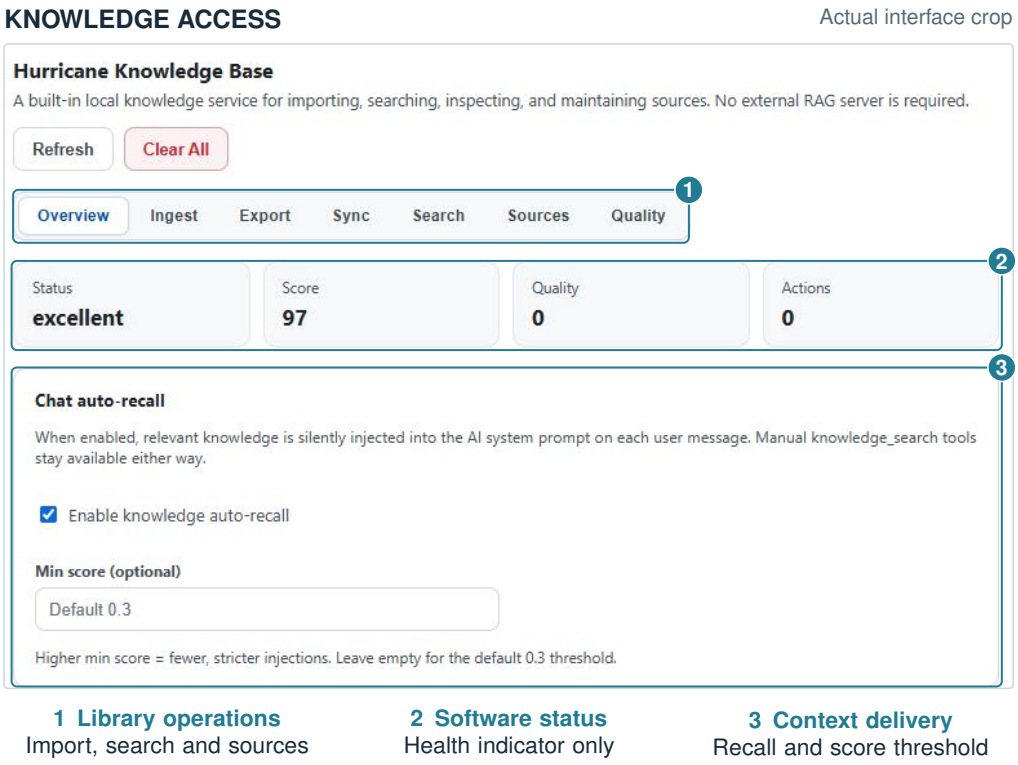


Figure 5: Hurricane's knowledge-library panel, cropped to library and recall controls. Numbered annotations identify library operations, software status, and automatic recall with a minimum-score setting. The displayed score of 97 is a software health/status indicator, not benchmark accuracy.

Workflow and development workspaces. The workflow view lets users select a saved workflow, supply task text or JSON, and access an editor; the result area is reserved for run status, node records, and human confirmation. The development view exposes project and technology settings, parallel-agent and repair-round limits, and a staged path from requirements through design, decomposition, conflict checking, implementation, merge, compilation, testing, and documentation. Separate panes display tasks and dependencies, agents, and an event timeline. The screenshots show idle configuration states, not successful runs of these workflows.

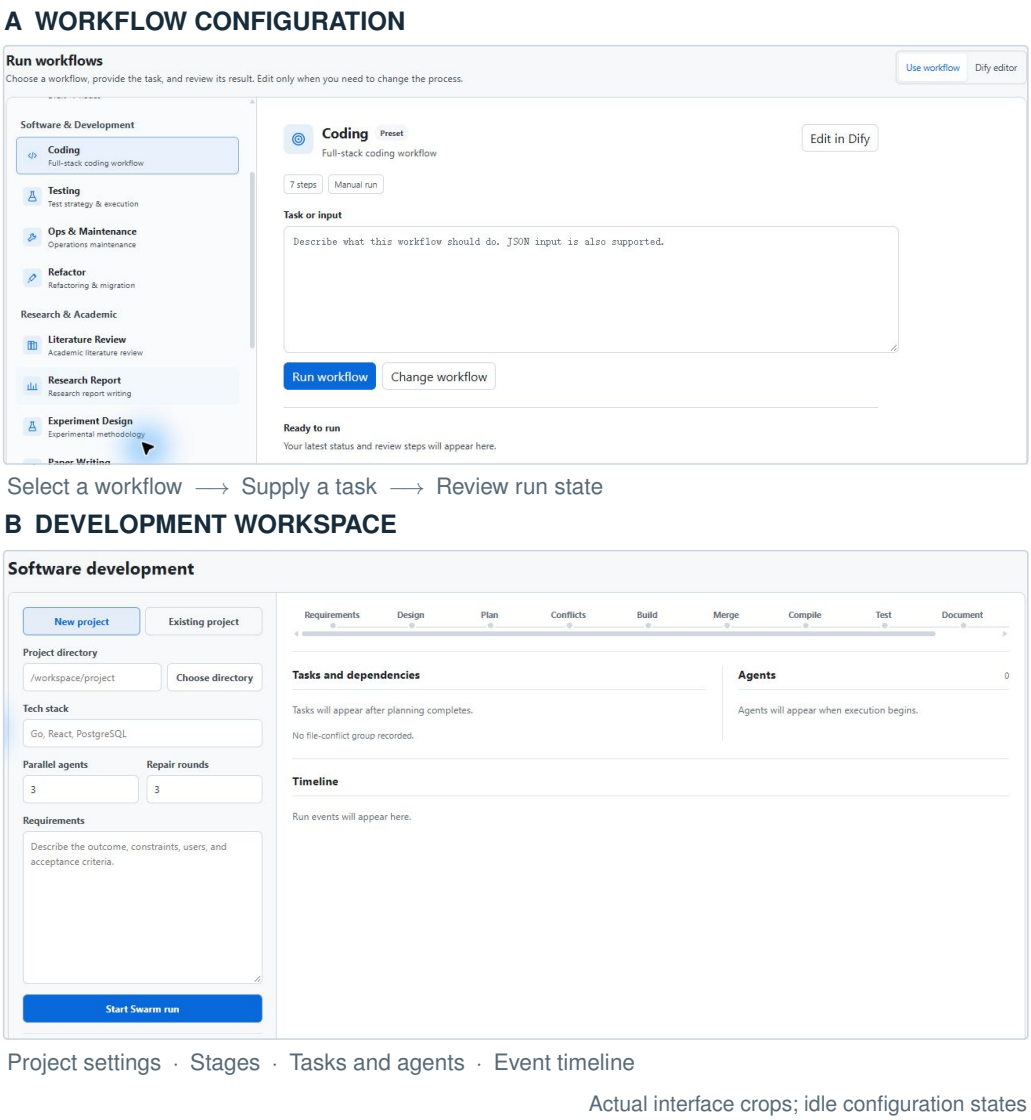


Figure 6: Hurricane workflow and development controls in idle configuration states. These interface views provide implementation context, not evidence of end-to-end task performance.

J Operational Interfaces and Execution Paths

Figure 7 expands Figure 1 into four functional modules: cross-task experience access, same-task state construction, reader execution, and post-execution organization. The host selects the path through task-specific adapters. The studies compare selected components with sources matched within each comparison; they do not evaluate a joint policy combining cross-task memory, same-task state, and online writeback.

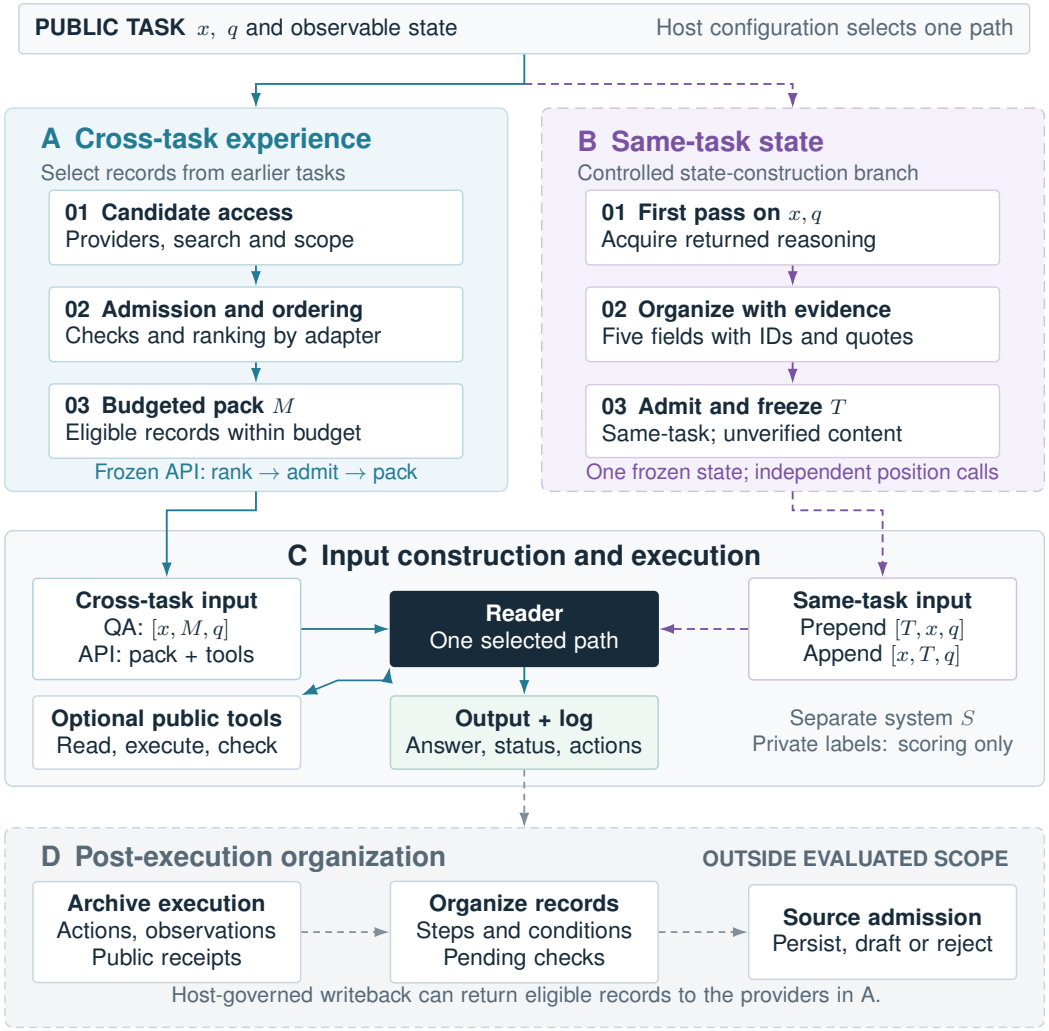


Figure 7: Functional architecture and evaluation scope. Blue shows cross-task access and reader execution; dashed purple denotes the separate same-task experiment. The two inputs in C are alternatives selected by configuration. D denotes online organization and writeback after target execution, outside the evaluated scope. Organization of frozen sources and same-task state is evaluated separately. Module B depicts the earlier partial-run long-context adapter (Appendix A.3); its five-list format is adapter-specific. Source checks establish traceability; reuse-time checks assess explicit conditions against public state. The general provider interfaces and frozen experimental adapters have distinct ordering and budget rules, detailed below.

J.1 Host configuration and experience access

The organizer generates record content; host configuration fixes its delivery order. In module A, the desktop application's composite experience provider combines memory, skill, and governance-draft providers through listing, search, and utility-update interfaces. Returned candidates include relevance, priority, scope information, estimated tokens, and reasons. This composition does not require two separate vector databases. Utility updates are provider-specific; the skill provider's update is currently a no-op. These software interfaces are not an end-to-end experimental result.

Distinct adapters. The QA adapter organizes fixed source experience for a target. The archived QA implementation places that experience after the public context and before the question. The API adapter instead evaluates historical code against current public contracts. Its frozen experiment first fixes a common Hybrid ranking, combining lexical and vector rankings by reciprocal rank fusion with denominator offset 60. It then filters candidates and preserves their relative order during whole-record packing. The general Go pack builder performs admission and conflict handling before ranking; it is a different entry point and is not substituted for the frozen experiment.

Same-task state. Figure 8 expands modules B and C. The earlier partial-run adapter (Appendix A.3) retains the first 50,000 characters of the returned reasoning field, divides them into 2,000-character events, and assigns event identifiers. The organizer emits five lists with at most four entries per list. Each admitted entry needs nonempty text, known evidence IDs, and a quotation of at least ten characters present in a cited event. The compatibility adapter fills missing lists with empty lists but rejects unknown fields. With no admitted entry, organization fails. The organizer receives neither the separately returned final-answer field nor scoring labels; the reasoning field may still contain unverified candidate answers.

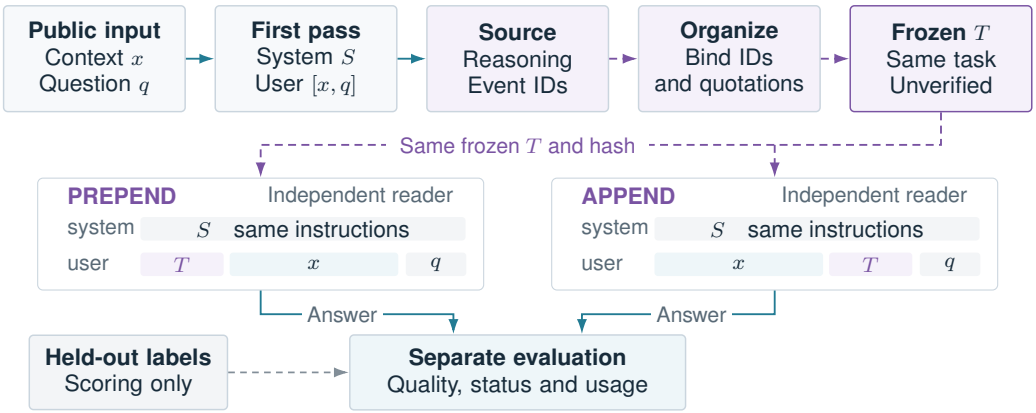


Figure 8: The earlier partial-run long-context branch. Dashed purple arrows denote state construction and controlled delivery, not a default desktop execution path. One frozen state supplies both independent calls. Hidden labels enter only the evaluator; hashes establish input identity, not semantic correctness.

Freezing, budgets, and failures. The adapter writes source, organizer-response, and state hashes before the position calls. The wrapper retains `<first_run_reasoning_traces>` and `[Trace 1]` markers with an unverified-state warning. All state stays in the user message; the question remains last. A state exceeding 50,000 characters is rejected without cutting its JSON. The separate representation diagnostic instead applies a 4,096-token wrapper-inclusive cap: structured state loses whole entries, while raw/prose text is truncated by tokens. Failure and transport-retry policies remain batch-specific (Appendices A.3 and H). These implementation limits are not inferred specifications of the completed three-model aggregate study.

J.2 Admission decisions and host responsibilities

Figure 9 summarizes reuse-time admission. The gate receives a record contract, current public state evidence, and host-supplied validation receipts. The contract binds the record ID, version, body digest, dependency digests, applicability clauses, and required checks. State evidence supplies observed file digests, an explicit dependency graph, and public facts. Dependency traversal checks both missing observations and undeclared transitive dependencies; leaf nodes must explicitly have no outgoing dependencies. Applicability compares public fact values against allowed values, without a semantic judge.

Route	Representative trigger	Host responsibility
`inspect`	Missing evidence, binding mismatch, or incomplete dependency closure	Obtain public evidence before re-assessment
`adapt`	Declared dependencies changed without an earlier applicability rejection	Construct an adapted candidate and updated bindings
`validate`	A required current receipt is missing or invalid	Execute the specified public check and record its result
`reject`	Scope failure or incompatible public pre-conditions	Withhold this record for the current request
`reuse_candidate`	Applicable, dependency-bound record with current required receipts	Make the record eligible for historical access

Table 17: Gate outputs and their interpretation. Decisions include reasons and missing, changed, or pending evidence. Routes describe host responsibilities; the frozen API experiment automatically services only the validation route.

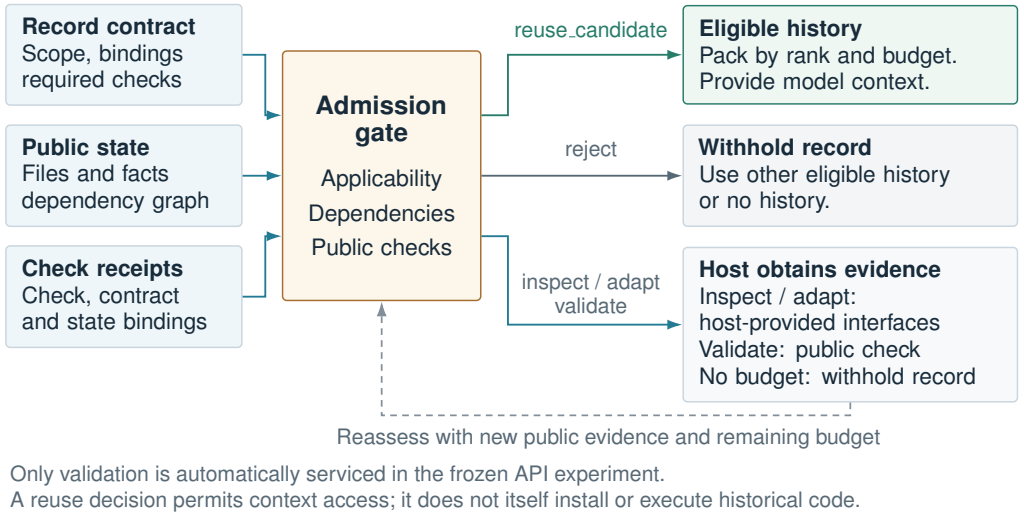


Figure 9: Reuse-time admission and host responsibilities. Public contracts, state evidence, and check receipts determine eligibility. The repair routes specify host interfaces; only validation is automatically serviced in the frozen API experiment. A reuse decision permits historical context access without establishing task success.

Receipt validity. A qualifying receipt matches the check ID and definition digest, current contract digest, and current state digest. It records a pass with at least one executed test, a completion time no later than the current time, and an unexpired interval ending after completion. A digest binds contents; it is not a signature or proof that a check actually ran. The host must provide the execution record. The API experiment reuses qualifying source-time receipts or executes public checks in an isolated container, then reassesses the candidate. It restores the initial solution and separates validation traces from the reader's action log before target execution.

The frozen API procedure. In the local API experiments, the adapter services validation within the target tool allowance. It provides neither an automatic inspect/adapt loop nor access to hidden tests.

1. Read the shared, pre-ranked archive and construct public state evidence.
2. Assess each record. If validation is required, use a qualifying cached receipt or run the current public check; assess again with that receipt.
3. Retain only `reuse_candidate` records for Hurricane and the matched log. Hybrid retains the unfiltered ranked archive.
4. Restore the target workspace. Greedily pack complete records into the initial message under the configured history cap (4,096 tokens in the main policy comparison). Populate the history tools according to the cohort's access rule below.
5. Run the reader with the remaining target budget. Record admission, packed IDs, initial messages, subsequent actions, submissions, and resolved usage.

In the main 36-task policy comparison, the condition-specific history store may contain more records than the initial pack. The reader can request a record through `work_history` or copy code through `work_reuse`; Hurricane and the matched log expose only admitted records. In the twelve-task matrix, every condition instead restricts both the initial input and later history tools to the same selected records. Records outside that selection are inaccessible throughout execution; top-1 selects at most the highest-ranked record. Tool returns contribute to later model input and target cost. Gate acceptance itself neither writes code nor establishes task success.

Validation accounting in the local API experiments. Source generation and source-time validation are preparation costs, allocated as specified for each cohort. Target admission checks run before the first target model call but consume the target's 600-tool-second allowance. These checks are programmatic and make no model calls. Public checks during target execution consume the same tool allowance; any model call used to request a check, and its tokens, also count toward the target model budget. Hidden grading is recorded separately, never returned to the agent, and excluded from its tool allowance. These accounting rules describe the local API experiments; a common numerical cap alone does not establish identical cost allocation in other execution batches.

Concrete contract example. The decimal task declares that `api_contract.json` depends on `options.json`. An experience that omits the options dependency is routed to inspection. If its public precondition permits only half-up rounding, a target requiring half-even is rejected: rounding `2.345` to two places changes from `2.35` to `2.34`. This applicability check precedes the changed-dependency adaptation route. If the contract and dependencies match but the receipt is stale, validation is requested instead. These are implemented decision cases, not additional performance measurements.

K Admission Decisions and Delivered Evidence

This post-hoc audit links all 36 targets in Table 3 to initial history packs, admission decisions, public checks, and outcomes.

The realized intervention. Hurricane admits and packs one record on 15 targets and none on 21. Each retained record is first in Hybrid's initial pack. Hurricane's history-tool store contains only its admitted record, or is empty when no record is admitted. Admission therefore combines history reduction with conditional retention. The matched log reproduces all 36 initial inputs. Record identity does not supply unobserved top-1 answers or costs.

	H history	Success count				Target tokens	
Scenario	exposed	H	Hybrid	Log	None	H	Hybrid
Stable conditions	5/6	6/6	6/6	6/6	6/6	19,853	55,425
Missing dependency	0/6	5/6	6/6	5/6	5/6	27,366	71,362
Transitive change	0/6	5/6	6/6	4/6	4/6	28,119	71,662
Applicability change	0/6	5/6	4/6	4/6	5/6	34,191	88,512
Expired receipt	5/6	6/6	5/6	6/6	5/6	24,633	75,750
Insufficient coverage	5/6	5/6	5/6	5/6	5/6	28,377	88,809
Total	15/36	32/36	32/36	30/36	30/36	162,539	451,520

Table 18: Post-hoc decomposition of the existing 36-target API cohort. H denotes Hurricane; Log uses the same checks; None has no history. Each scenario row contains six targets, with one execution per target and condition. Tokens sum recorded target model use, including failures; source acquisition and tool costs are excluded here. These small groups describe the cohort, not independent estimates of component effects.

Public evidence behind decisions. Across 216 candidate assessments, the terminal routes are 162 rejections, 36 inspections, 15 reuse candidates, and three pending validations. Archived public family/mode values corroborate the rejections; the dependency graph corroborates the omitted `options.json` declarations. Separate pre-injection logs record 18 public checks covering 27 examples; comparing observed values with frozen expected outputs gives 15 passes and three failures. All three failures involve the CSV source in the stable, expired-receipt, and coverage scenarios: its code supplies `comma` as a CSV delimiter and the execution records a `TypeError`. This also explains the stable group's empty pack.

Repeated reader inputs. Stable/expired-receipt and transitive-change/applicability pairs have identical initial messages within each family and four arms: 48 byte-identical pairs. All 12 reconstructable cached receipts had expired before admission; both stable and expired-receipt cases ran current checks. Transitive-change and applicability cases both terminate at applicability; none reaches adaptation. These labels do not isolate independent dependency and applicability effects.

Outcomes by historical access. Hurricane and Hybrid both solve 17/21 withheld-history targets, with two wins and two losses; their target-token totals are 112,370 and 316,749. Both solve all 15 retained-history targets, using 50,169 and 134,771 tokens. These post-hoc strata are defined by Hurricane's admission, not randomized groups. On all 21 withheld-history targets, Hurricane and no history have byte-identical initial messages, yet solve 17/21 and 15/21; that difference cannot be attributed to different initial historical context.

Audit boundary. This is a consistency check using archived public evidence, not a new execution or a semantic ground truth. Complete receipt payloads were not persisted; exact receipt binding cannot be independently verified. Public check logs lack candidate IDs and body hashes, so their association relies on the frozen procedure and unique compatible candidate. The audit locates the intervention but does not establish superiority over top-1, a harmful-transfer rate, or admission precision/recall.

L Admission Cases from the Direct-Execution Conditions

This section analyzes the direct-execution runs in Table 10: Hurricane succeeds on 31/36 targets (86.1%), compared with 28/36 (77.8%) for Hybrid. The aggregate table and the cases below describe the same executions. This cohort is separate from the 32/36–32/36 policy comparison in Table 3 and the twelve-task mechanism comparison in Table 4. "Direct execution" names the execution condition, not a no-history baseline: Hybrid can use the historical implementation, whereas Hurricane controls its delivery. The following three CSV cases explain execution trajectories within this cohort; they are not an additional independent replication.

Case	Hurricane admission decision	Hybrid execution failure
CSV T2	Missing binding to `options.json`; dependency check withholds history.	Copies the old code and submits; using `"comma"` as the delimiter raises `TypeError`.
CSV T4	Target requires a pipe delimiter, but history specifies a comma; applicability check withholds history.	Repeated attempts to repair delimiter handling exhaust the call allowance without submission.
CSV T6	Old code fails checks on two public examples; execution validation withholds history.	Repairs the delimiter but retains `splitlines()`, losing newlines within quoted fields; private grading detects the defect.

Table 19: Three CSV cases from the direct-execution conditions in Table 10. Hurricane succeeds in each after reading the current configuration and rewriting the implementation. Only T6 invokes execution validation; its public checks detect the delimiter error, not the quoted-newline defect exposed by private grading.

From admission to target execution. The cases exercise three different parts of the record-and-admission interface. In T2, the historical implementation lacks a binding to `options.json`, so dependency admission withholds it. In T4, the target requires a pipe delimiter while the historical conditions specify a comma, so applicability admission withholds it. In T6, execution against two public examples rejects the historical implementation. In all three cases, Hurricane reads the current configuration, writes a new implementation, and succeeds; Hybrid fails after copying or attempting to repair the earlier implementation. These observations connect explicit reuse conditions to changed historical access and subsequent execution.

What validation detected. Among these three cases, only T6 invokes execution-based validation. Its public checks detect a delimiter error in the old implementation. Hybrid subsequently repairs that error but retains `splitlines()`, which removes newlines inside quoted fields; this defect is exposed by private grading. The public validator therefore does not diagnose the later newline defect. Hurricane's successful rewrite follows rejection of an implementation that fails the public checks, rather than evidence that those checks detect every defect.

Interpretation and scope. The aggregate difference is three successes (8.3 percentage points), with dependency, applicability, and validation represented among the three reported favorable cases. It is not a validation-only gain. The counts and examples do not specify the full paired outcome matrix; we do not infer that there are no other favorable or unfavorable differences. The cases illustrate admission behavior and possible transfer failure paths, without isolating each check's causal effect or estimating a general harmful-transfer rate. They concern one task family and do not identify which tasks account for the separate twelve-task validation-ablation gap.