COMPOSING EXPERIENCE ACROSS TASKS: HIERARCHICAL PROCESS EXPERIENCE MEMORY FOR INTERACTIVE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Interactive agents repeatedly face tasks that share reusable subprocesses, but memories based on complete trajectories, task-specific reflections, or standalone routines do not always preserve how subprocesses combine or whether locally plausible guidance helps the final task. We introduce Hierarchical Process Experience Memory (HPEM), an external plug-in memory that organizes experience abstracted from concrete action arguments into leaf processes and composed workflows. Each node records an objective, interaction stage, subgoals, applicability and failure boundaries, evidence, and maturity; composed workflows additionally store ordered references to shared child nodes. Through benchmark-specific adapters, HPEM acquires process abstractions from successful interactions, composes compatible experience into higher-level workflows, and updates node reliability using both local feedback and downstream task outcomes. At inference time, HPEM retrieves a relevant leaf process or composed workflow, or temporarily composes stored processes, as optional guidance based on the current process need; the host agent retains full control of reasoning and environment actions. Memory is frozen before evaluation to separate experience acquisition from cross-task reuse. Extensive experiments show gains in task success and interaction efficiency in multiple settings with official host agents, while frozen evaluations document retrieval of learned high-level workflows.

1 INTRODUCTION

Interactive agents encounter tasks that share subprocesses: locating a target, satisfying a prerequisite, resolving ambiguity, and verifying a change. A process may complete one task yet serve as an intermediate step in another. Reuse therefore requires preserving applicability, dependencies, and expected outcomes alongside the process itself.

Memory architectures retain trajectories, reflections, workflows, and skills as complementary experience (Sumers et al., 2023; Zhang et al., 2025b; Zheng et al., 2024; Shinn et al., 2023; Wang et al., 2024; 2023). We study a specific representation question: how can an advisory memory preserve the identity of a subprocess shared by several ordered workflows? Persistent identity attaches subsequent evidence to one node rather than separate textual copies. Local verification records progress, while attributed task outcomes associate references with final outcomes.

Hierarchical Process Experience Memory (HPEM) represents leaf processes and composed workflows in a persistent directed acyclic graph. Both use the same schema for intent, description, evidence, and maturity. Workflows store ordered references to existing children; multiple parents can share a child, which can itself be composed.

During acquisition, benchmark-specific deterministic adapters map successful interactions to process observations and consolidate workflows. The semantic vocabulary and mapping rules are authored; the stored nodes, combinations, and evidence are accumulated from interaction. Local verification and downstream task outcomes inform node reliability and retrieval eligibility. Before evaluation, memory is frozen. HPEM then supplies optional process references while the existing host retains responsibility for reasoning and every environment action. Figure 1 illustrates reference provision across an evolving interaction. Our contributions are:

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.

- **Persistent shared process composition.** A common node schema and ordered parent–child relations preserve constituent processes and associate evidence with shared identities across workflows.
- **Evidence-driven advisory memory.** Local and downstream evidence govern process eligibility under an acquisition–freeze–reuse protocol that preserves the host's action interface.
- **Cross-task reuse and cross-model transfer.** Paired evaluations demonstrate task-success gains across multiple host agents and benchmarks. Matched-acquisition comparisons establish improved strict completion over flat process retrieval and higher reward with lower execution cost than trajectory retrieval. An online ablation evaluates the contribution of downstream feedback to memory construction and reuse.

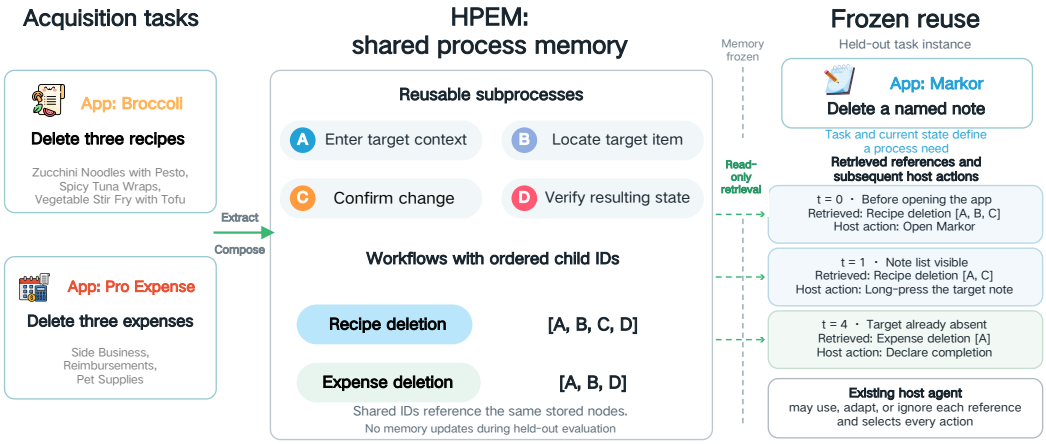


Figure 1: **Recorded cross-task reference provision.** Recipe and expense deletion share persistent nodes A (enter target context), B (locate target item), and D (verify resulting state); C denotes confirmation. Ordered child IDs reference the same stored subprocesses. During a held-out Markor instance, the frozen memory provides recipe references at host steps $t = 0, 1$ and an expense reference at $t = 4$, according to the current process need. The timeline pairs recorded references with subsequent host actions. The host may use, adapt, or ignore each reference and selects every action; memory remains unchanged throughout evaluation. Held-out denotes a new task instance rather than an unseen task template (Appendix A.3).

2 Related Work

2.1 Trajectory Retrieval and Reflective Memory

Experience memories differ in what they retain from prior interactions. Synapse retrieves abstracted state–action trajectories as exemplars (Zheng et al., 2024). Reflexion converts feedback into verbal reflections (Shinn et al., 2023), while ExpeL extracts cross-task insights and retrieves successful experiences (Zhao et al., 2024). ReasoningBank distills reusable reasoning strategies from successful and failed interactions (Ouyang et al., 2026). Buffer of Thoughts maintains reusable thought templates that are retrieved and instantiated for subsequent problems (Yang et al., 2024). AutoManual constructs instruction manuals from environmental interactions, organizing accumulated rules to guide subsequent planning (Chen et al., 2024).

HPEM makes the shared subprocess an explicit memory unit with applicability conditions and failure boundaries. Workflows reference persistent constituents, so later observations and task outcomes update the same identity's evidence, reliability, and maturity.

2.2 Reusable Workflows and Skill Composition

Agent Workflow Memory induces reusable routines and progressively more complex workflows (Wang et al., 2024). AFlow automates agentic workflow generation through search over code-represented workflows, using execution feedback to guide optimization (Zhang et al., 2025a). LifeMem clusters trajectories by workflow to extract reusable skills (Qiu et al., 2026). Feng et al. compare task- and subtask-level skill induction and text and code formats (Feng et al., 2026b).

Executable skill libraries provide another route to experience reuse. Voyager maintains composable executable skills (Wang et al., 2023), while DynaSaur generates actions as programs and accumulates them for subsequent reuse (Nguyen et al., 2024). BOSS expands a learned skill library through LLM-guided bootstrapping, practicing skill sequences to acquire longer-horizon behaviors (Zhang et al., 2023). HiSkill represents hierarchical skill relations (Hao et al., 2026), while GraSP organizes dependency-aware skill execution with verification and repair (Xia et al., 2026). Graph-of-Skills retrieves dependency-aware bundles from existing skill packages (Liu et al., 2026); CaSKG calibrates procedural relations through counterfactual probes (Li et al., 2026b). SkillGraph jointly evolves a skill graph and an agent policy through reinforcement learning (Li et al., 2026a).

HPEM combines persistent identities, ordered composition, and an advisory interface. Workflows share child nodes and their accumulated evidence. References describe what to accomplish and verify; the host interprets them and generates actions. This couples shared representation and node-level evidence without replacing host execution.

2.3 Graph-Structured Experience and Outcome Feedback

A-MEM creates structured memory notes, dynamically links related notes, and updates their contextual representations as new memories arrive (Xu et al., 2026b). G-Memory organizes multi-agent experience through a hierarchy of insight, query, and interaction graphs, retrieving both generalizable insights and condensed collaboration trajectories (Zhang et al., 2026). UI-Mem organizes workflows, subtask skills, and failure patterns for mobile GUI agents and integrates memory guidance with online reinforcement learning (Xiao et al., 2026).

EXG organizes successful and failed experiences into a graph for online and offline reuse (Jin et al., 2026). ExpGraph connects distilled skills and failure lessons and incorporates downstream utility into retrieval (Feng et al., 2026a). HyperSkill links subtask steps and reusable skills through trajectory hyperedges, with dual-path retrieval and structure-informed maintenance (Xu et al., 2026a).

HPEM gives its hierarchy a specific procedural meaning: parent–child relations encode ordered composition, multiple workflows may reference the same child, and children may themselves be composed workflows. Local verification and downstream task outcomes inform node reliability and maturity during acquisition. The focus is a shared process representation whose constituent identities and evidence persist across workflows. We evaluate an advisory artifact after freezing it, with the host policy unchanged, rather than training a policy to internalize memory-guided behavior.

3 Hierarchical Process Experience Memory

HPEM couples an external process memory to an existing host agent. At interaction step t, a deterministic adapter maps the task and current state to a *process need* q_t: domain, objective, interaction stage, task family, subgoals, and context tags. Process guidance abstracts concrete action arguments; context tags may retain task descriptors. HPEM retrieves guidance for the host's interaction history h_t; the host chooses the next action a_t.

The memory graph $G = (V, E)$ grows and accumulates evidence during acquisition. Frozen evaluation permits retrieval and rendering without persistent updates (Figure 2).

3.1 Process Experience Representation

Each node $v \in V$ is represented by $v = (z_v, p_v, c_v, r_v, e_v, m_v, \ell_v)$. Its intent z_v shares the fields of q_t. The description p_v contains a title, abstract steps, expected success signals, strategy tags, applicability conditions, and failure boundaries. The ordered child-ID sequence c_v has correspond-

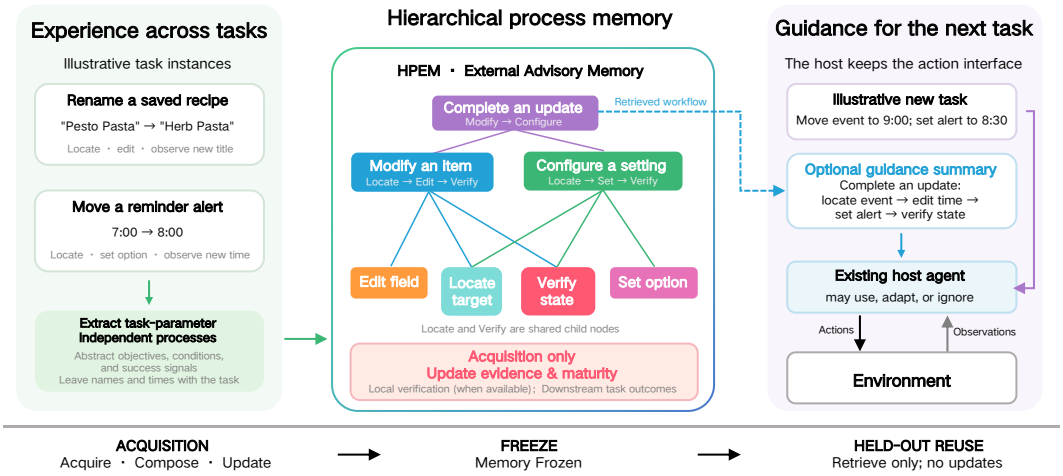


Figure 2: **HPEM lifecycle.** Deterministic adapters abstract successful interactions into leaf processes and consolidate ordered workflows with shared children. Local verification and downstream outcomes update evidence, reliability, and maturity. After freezing the graph, node states, and retrieval configuration, HPEM renders optional guidance. The host retains its action interface. The tasks and hierarchy are illustrative.

ing child-role and composition descriptions r_v. Let $\mathcal{C}_v \subseteq V$ denote the nodes referenced by c_v. Evidence e_v, maturity m_v, and level ℓ_v describe the node's accumulated state.

A leaf has no children, while a composed workflow references existing children in the order stored in c_v. Its level is

$$\ell_v = \begin{cases} 1, & \mathcal{C}_v = \emptyset, \\ 1 + \max_{u \in \mathcal{C}_v} \ell_u, & \mathcal{C}_v \neq \emptyset. \end{cases} \tag{1}$$

Composition edges are directed from parent to child: $E = \{(v, u) : v \in V,\ u \in \mathcal{C}_v\}$. Leaves occupy level 1, and composed or *high-level* workflows occupy levels 2 and above. Validation checks child existence, level consistency, and acyclicity. Multiple parents may reference the same child identity, yielding a hierarchical DAG.

3.2 Learning and Composing Processes

Benchmark-specific adapters define the semantic vocabulary, state/action mappings, verification rules, and text templates. Successful interactions supply process variants and child combinations; local verification and attributed task outcomes update their evidence, without separate model calls. Appendix A.1 specifies the mappings.

Leaf observations are matched within exact semantic-intent scopes and reinforced according to strategy-tag similarity; unmatched observations can create bounded variants. A persistent workflow requires at least two distinct valid child IDs. Composition is triggered by a successful workflow containing learned children, an offered temporary composition containing credited children, or explicit adapter consolidation of a successful workflow.

Child IDs are deduplicated in adapter- or trace-provided first-occurrence order. Workflow variants match within the same intent scope and level using order-insensitive child-profile similarity; rendering preserves order. Thus, child IDs describe a composition of distinct components, not a complete action trace or explicit loop. The host may repeat actions when instantiating that guidance. The fixed matching rules, variant limits, and reinforcement priorities are specified in Section A.

3.3 Outcome-Aware Evidence and Maturity

The evidence record is $e_v = (O_v, N_v^{\mathrm{L}}, S_v^{\mathrm{L}}, N_v^{\mathrm{D}}, S_v^{\mathrm{D}})$. Here, O_v counts successful process observations; N_v^{L} and S_v^{L} count local-use trials and successful local verifications, respectively; N_v^{D} and S_v^{D}

count downstream-attributed outcomes and successful outcomes. All counts are nonnegative integers, with $S_v^{\mathrm{L}} \le N_v^{\mathrm{L}}$ and $S_v^{\mathrm{D}} \le N_v^{\mathrm{D}}$. Task-ID deduplication uses a bounded history (Section A.1).

Reliability prioritizes repeated local verification, then repeated downstream evidence, and otherwise pools sparse evidence:

$$\rho_v = \begin{cases} \dfrac{S_v^{\mathrm{L}} + 1}{N_v^{\mathrm{L}} + 2}, & N_v^{\mathrm{L}} \ge 2, \\ \dfrac{S_v^{\mathrm{D}} + 1}{N_v^{\mathrm{D}} + 2}, & N_v^{\mathrm{L}} < 2 \ \wedge \ N_v^{\mathrm{D}} \ge 2, \\ \dfrac{S_v^{\mathrm{L}} + S_v^{\mathrm{D}} + 1}{N_v^{\mathrm{L}} + N_v^{\mathrm{D}} + 2}, & \text{otherwise.} \end{cases} \tag{2}$$

Without trials, $\rho_v = 0.5$. The rule prioritizes local subprocess verification, using downstream outcomes when local evidence is sparse. It is an operational score; O_v enters maturity and retrieval separately. In the controlled Full snapshot, substituting pooled reliability preserves all nine maturity labels (Appendix A.3).

For maturity decisions, let A_v and B_v denote the effective attempt and success counts derived from the two evidence channels:

$$(A_v, B_v) = \begin{cases} (N_v^{\mathrm{L}}, S_v^{\mathrm{L}}), & N_v^{\mathrm{L}} \ge 2, \\ (N_v^{\mathrm{D}}, S_v^{\mathrm{D}}), & N_v^{\mathrm{L}} < 2 \ \wedge \ N_v^{\mathrm{D}} \ge 2, \\ (\max\{N_v^{\mathrm{L}}, N_v^{\mathrm{D}}\}, \max\{S_v^{\mathrm{L}}, S_v^{\mathrm{D}}\}), & \text{otherwise.} \end{cases} \tag{3}$$

Equation 2 smooths the ranking evidence, whereas Equation 3 gates eligibility. Its sparse maxima prevent one local and one downstream record from supplying two validations solely because they occupy different channels.

A node is first marked *retired* if $A_v \ge 6$ and $\rho_v < 0.35$. Otherwise, it is *trusted* if $O_v \ge 2$, $B_v \ge 2$, and $\rho_v \ge 0.6$; remaining nodes are *candidates*. These are fixed eligibility rules. Reference-use credit is restricted to offered IDs: a direct workflow offers its root ID, while a temporary reference offers its selected roots. Rendering children does not automatically credit every descendant. Multiple credited roots may share the same task outcome; alignment records association, not causal adoption (Appendix A.1).

3.4 Retrieval and Host-Preserving Assistance

Retrieval selects trusted roots in the current domain. Semantic strings are stripped and lowercased; subgoals and context tags are deduplicated in first-occurrence order.

Define $\Phi(q_t, v) = (I_o, I_s, I_f, H_g, J_c)^\top$, with all features evaluated for the pair (q_t, v). I_o indicates objective equality. I_s and I_f equal one only when the request's stage or task-family field, respectively, is nonempty and equals the node's corresponding field; otherwise they are zero.

Let $\mathcal{S}_t$ and $\mathcal{S}_v$ denote the requested and stored subgoal sets. Requested-subgoal coverage is $H_g = |\mathcal{S}_t \cap \mathcal{S}_v| / \max(1, |\mathcal{S}_t|)$. Let $\mathcal{T}_t$ and $\mathcal{T}_v$ denote the corresponding context-tag sets. Their Jaccard overlap is $J_c = |\mathcal{T}_t \cap \mathcal{T}_v| / |\mathcal{T}_t \cup \mathcal{T}_v|$ when the union is nonempty, and $J_c = 0$ otherwise. Semantic fit is

$$\sigma(q_t, v) = \mathbf{w}^\top \Phi(q_t, v), \qquad \mathbf{w} = (0.38, 0.16, 0.08, 0.30, 0.08)^\top. \tag{4}$$

Equation 4 is a hand-specified structured matcher. With evidence quality $Q_v = \rho_v \min(1, O_v/2)$, the final ranking score is

$$R(q_t, v) = 0.72\, \sigma(q_t, v) + 0.28\, Q_v + 0.08\, \mathbf{1}[m_v = \text{trusted}], \tag{5}$$

where $\mathbf{1}[\cdot]$ is the indicator function. The weights are fixed. Trusted roots have $Q_v = \rho_v$, a constant maturity bonus, and $0.248 \le R \le 1.08$, so the 0.18 threshold does not filter them. The top-six shortlist follows R, breaking ties by experience ID. R is a ranking score, not a probability, and includes no hierarchy-level bonus.

HPEM selects the highest-ranked exact-objective root covering all requested subgoals. A leaf supplies its description. For a workflow, rendering recursively retains children whose objectives or

subgoals intersect the request; if none match at a node, it retains that node's full child list. Stored order is preserved. Eligibility is checked at the root, not again at descendants; child retirement does not invalidate a parent automatically.

Otherwise, the composer assigns a distinct qualifying root to each requested subgoal in order. A temporary reference requires complete coverage and at least two roots, without creating a persistent node. Exact qualification and empty-request rules are in Appendix A.1.

3.5 Frozen Memory and the Host Boundary

Let π_{H} denote the existing host policy. Optional guidance γ_t is added to its interaction history h_t:

$$\gamma_t = \mathrm{Retrieve}(q_t, G), \qquad a_t \sim \pi_{\mathrm{H}}(\cdot \mid h_t, \gamma_t). \tag{6}$$

Here, $\mathrm{Retrieve}$ includes reference selection and rendering. If no reference qualifies, γ_t is empty and the host proceeds with its existing context. The host may use, adapt, or ignore the guidance and selects every environment action.

Frozen evaluation fixes the graph, node contents, evidence, maturity, and retrieval configuration. Retrieval and temporary rendering remain available; evaluation logs are separate from memory updates (Appendix A.3).

4 Experiments

We evaluate frozen cross-task reuse, memory representations under matched acquisition data, and downstream feedback during acquisition. Cross-model comparisons and structural audits further assess artifact reuse and shared workflow retrieval.

Table 1: WebShop effectiveness with the 8B backbone on 500 paired goals per host. Δ is HPEM minus Host, in pp for success and partial success and benchmark units for reward.

Host	Metric	Host	+HPEM	Δ [95% CI]
Native	Success (%)	8.2	14.8	**+6.6 [4.0, 9.2]**
	Reward	0.191	0.294	**+0.103 [0.074, 0.131]**
	Partial (%)	26.8	38.8	**+12.0 [8.4, 15.4]**
ReAct	Success (%)	15.4	29.0	**+13.6 [9.8, 17.4]**
	Reward	0.304	0.493	**+0.189 [0.150, 0.229]**
	Partial (%)	44.2	61.8	**+17.6 [12.6, 22.6]**
Reflexion	Success (%)	21.6	30.4	**+8.8 [5.0, 12.8]**
	Reward	0.282	0.306	+0.024 [−0.015, 0.064]
	Partial (%)	33.2	30.8	−2.4 [−6.8, 2.0]

4.1 Experimental setup

Benchmarks and acquisition. The primary WebShop (Yao et al., 2022a) experiments use the official shuffled 1,000-product, 6,910-goal profile, with 300 acquisition goals and 500 disjoint goals for paired frozen-memory comparisons. Native, ReAct (Yao et al., 2022b), and Reflexion (Shinn et al., 2023) each acquire a separate memory artifact. An additional ExpeL (Zhao et al., 2024) comparison with Qwen3.5-27B uses 500 paired goals.

We compile the ALFWorld (Shridhar et al., 2020) memory from 24 successful training trajectories—four from each of the six task families. The resulting memory contains 28 trusted level-1/2 nodes. All acquisition task IDs are disjoint from the complete set of 134 `valid_unseen` evaluation instances. Both AdaPlanner (Sun et al., 2023) models read the same frozen artifact. AndroidWorld (Rawles et al., 2025) uses M3A with 116 acquisition task classes followed by 116 frozen paired evaluation classes.

Backbones, controls, and metrics. We use Qwen3-VL-8B-Instruct (Bai et al., 2025) as the primary backbone for experiments across benchmarks and host agents. Additional comparisons with

Qwen3.5-27B examine experience reuse with a different backbone, including reuse of the 8B-acquired ALFWorld memory. Within each Host versus Host+HPEM comparison, both arms use the same backbone, task instances, host configuration, and interaction budget (Section A.2). WebShop strict success is defined as $\text{reward} \geq 1$ and partial success as $\text{reward} > 0$. ALFWorld reports task success, while AndroidWorld reports the mean official task score, including fractional credit. Differences in percentage-valued metrics are reported in percentage points (pp); reward differences retain the benchmark's original scale. Paired bootstrap intervals quantify evaluation-unit variation conditional on each frozen artifact and configuration (Appendix A.3). Throughout the tables, bold difference intervals exclude zero; metric direction determines whether a difference is favorable.

Interaction cost. WebShop accounting includes all attempts and reflection calls; API token totals include prompts, completions, and memory guidance. HPEM performs abstraction and retrieval without separate model calls. ALFWorld reports model calls, tokens, and wall time across its planning and repair procedure. These measurements capture evaluation-time execution with an acquired artifact and exclude acquisition costs. AndroidWorld reports environment steps separately for acquisition and frozen evaluation.

Table 2: ALFWorld AdaPlanner on 134 paired tasks using the same 8B-acquired artifact. Panel (a): success differences in pp. Panel (b): descriptive per-task means; tokens include prompt and completion usage, latency is wall time, and acquisition cost is excluded.

(a) Task success

Model	Host	+HPEM	Δ [95% CI]
8B	15.7%	31.3%	**+15.7 [7.5, 23.9]**
27B	26.9%	44.8%	**+17.9 [9.7, 26.1]**

(b) Evaluation-time cost

Model	Arm	Calls	Tokens	Latency (s)
8B	Host	13.46	36,310	294.36
	+HPEM	12.73	28,107	231.53
27B	Host	12.81	34,116	580.42
	+HPEM	13.43	31,117	508.67

4.2 Main results: frozen cross-task reuse

HPEM improves strict task success across all three evaluated WebShop 8B hosts (Table 1): Native gains 6.6 pp, ReAct 13.6 pp, and Reflexion 8.8 pp. Native and ReAct also achieve higher reward and partial success. These results demonstrate successful integration with direct-action, reasoning-based, and reflection-based hosts while preserving their action interfaces.

On ALFWorld, the same frozen artifact improves both AdaPlanner backbones (Table 2): completed tasks increase from 21 to 42 at 8B and from 36 to 60 at 27B out of 134, yielding gains of 15.7 and 17.9 pp, respectively.

On AndroidWorld, the observed mean frozen-evaluation score rises from 35.8% to 39.2% ($\Delta = 3.5$ pp, 95% CI $[-3.5, 10.8]$; Table 3). The accompanying lifecycle audit records retrieval of learned high-level workflows during mobile interaction (Section 4.5).

4.3 Transfer and interaction efficiency

On ALFWorld, 27B AdaPlanner directly reuses the 8B-acquired memory without additional acquisition (Table 2).

HPEM also augments ExpeL, a host that already maintains experience. With Qwen3.5-27B, adding HPEM increases success from 61.8% to 63.2% and reward from 0.690 to 0.705 (Table 4). Among discordant task pairs, eight succeed only with HPEM and one only with the host baseline. The comparison retains ExpeL's existing memory and measures the incremental benefit of adding process guidance.

Table 3: AndroidWorld M3A on 116 paired task classes per phase. Δ is HPEM minus Host, in pp for official scores and actions for steps. Official scores permit fractional credit.

Phase	Host	+HPEM	Δ [95% CI]
(a) Mean official task score (%)			
Acquisition	35.8	39.7	+3.9 [−2.6, 10.3]
Frozen evaluation	35.8	39.2	+3.5 [−3.5, 10.8]
(b) Mean environment steps			
Acquisition	15.08	15.57	+0.49 [−0.66, 1.84]
Frozen evaluation	13.93	13.24	−0.69 [−2.34, 0.69]

Table 4: WebShop ExpeL with Qwen3.5-27B on 500 paired goals. Δ is HPEM minus Host, in pp for success and benchmark units for reward. Intervals are pointwise paired bootstrap 95% CIs.

Metric	ExpeL	+HPEM	Δ [95% CI]
Success (%)	61.8	63.2	**+1.4 [0.4, 2.6]**
Reward	0.690	0.705	**+0.014 [0.001, 0.029]**

On WebShop, ReAct and Reflexion combine higher strict success with fewer actions and tokens (Table 5). ReAct saves 1.11 actions and 1.92k tokens per episode; Reflexion saves 2.96 actions and 6.71k tokens. Native reduces actions by 1.26 per episode with an additional 0.68k tokens.

Table 5: WebShop evaluation-time cost on 500 paired goals per host. Δ is HPEM minus Host; lower is better. Acquisition cost is excluded.

Host	Cost	Host	+HPEM	Δ [95% CI]
Native	Steps	12.17	10.91	**−1.26 [−1.62, −0.89]**
	Tokens (k)	6.85	7.53	**+0.68 [0.40, 0.94]**
ReAct	Steps	11.05	9.94	**−1.11 [−1.54, −0.67]**
	Tokens (k)	18.97	17.05	**−1.92 [−2.77, −1.06]**
Reflexion	Steps	42.94	39.98	**−2.96 [−4.69, −1.29]**
	Tokens (k)	81.71	75.01	**−6.71 [−10.07, −3.51]**

On ALFWorld, both backbones combine higher success with lower mean token consumption and wall time (Table 2), demonstrating more efficient execution with the same frozen artifact.

4.4 Matched-acquisition memory comparisons

An additional controlled study uses official ReAct with the 8B backbone, 300 acquisition goals, and 500 frozen evaluation goals. All arms share a 15-step limit, a 3,000-character reference budget, and retrieval top-$k = 6$. *Flat process retrieval* uses the same graph as Full, restricting candidates to trusted leaves *before* ranking and truncation. *Trajectory retrieval* uses successful trajectories from the same acquisition records, without hierarchical composition. Thus, Full and Flat differ in their retrieval view of one graph; Trajectory changes the memory representation using the same source data. This study has its own Full condition (30.0%), distinct from the primary ReAct comparison (29.0%).

Strict completion. Full achieves 30.0% success versus 24.6% for Flat, a gain of 5.4 pp (95% CI [1.4, 9.4]; Table 6). Exact McNemar testing gives $p = 0.00934$, or $p = 0.0187$ after Holm correction over the two memory-baseline comparisons. Full and Flat share the same frozen graph and adapter vocabulary. Full achieves higher strict success with shorter mean references (884 versus 1,454 characters); Flat achieves higher reward and partial success at lower execution cost.

Reward and efficiency. Against Trajectory, Full increases reward by 0.072 and saves 0.56 steps and 2.52k tokens per episode; all three paired difference intervals exclude zero. Strict success is

Table 6: Controlled WebShop ReAct comparisons on 500 paired goals. Panel (a): success/partial success in percent, reward on the benchmark scale, and mean evaluation steps/tokens (k = 1,000). Full, Flat, and Trajectory share acquisition data; w/o feedback is acquired separately on the same goals. Panel (b): Full minus comparator, with paired 95% CIs; success differences are in pp. Bold intervals exclude zero.

(a) Frozen evaluation outcomes

Memory	Success ↑	Reward ↑	Partial ↑	Steps ↓	Tokens (k) ↓
Flat process	24.6	0.544	75.0	8.62	14.99
Trajectory	27.2	0.437	56.8	10.33	19.23
Full HPEM	30.0	0.509	63.8	9.77	16.71
w/o feedback	21.2	0.320	43.0	11.31	19.70

(b) Paired differences: Full minus comparator

Comparator	Success [95% CI]	Reward [95% CI]	Tokens (k) [95% CI]
Flat	**+5.4 [1.4, 9.4]**	**−0.036 [−0.066, −0.005]**	**+1.71 [1.06, 2.37]**
Trajectory	+2.8 [−1.0, 6.6]	**+0.072 [0.036, 0.109]**	**−2.52 [−3.25, −1.77]**
w/o feedback	**+8.8 [4.8, 13.0]**	**+0.189 [0.148, 0.230]**	**−2.99 [−3.82, −2.13]**

30.0% versus 27.2% ($\Delta = 2.8$ pp, 95% CI [−1.0, 6.6]). Appendix A.3 reports statistical details, reference lengths, and frozen-artifact audits.

4.5 Downstream feedback and memory construction

Both variants start from empty graphs and retain local process learning, local-use evidence, and composition; the ablation disables task-level downstream feedback. They acquire memory online separately using the same 300 goals, model, protocol, and budgets, measuring feedback's total effect on acquisition and frozen reuse.

Completion and efficiency. Full achieves 30.0% success versus 21.2% without feedback, a gain of 8.8 pp (95% CI [4.8, 13.0]; exact McNemar $p = 5.40 \times 10^{-5}$). Reward increases by 0.189, while steps and tokens decrease by 1.53 and 2.99k (Table 6).

Memory structure and coverage. The ablation retains 592 learning and five composition events. Full produces nine trusted nodes, including five high-level nodes, across four levels; the ablation produces two trusted high-level nodes across two levels. Memory is offered in 500 versus 246 evaluation episodes, linking feedback to memory maturity, hierarchy depth, and coverage. Additional component comparisons retain their protocol-specific Full artifacts (Appendix A.3).

AndroidWorld's frozen 43-node graph contains nine trusted high-level nodes with trusted descendants. Evaluation records 95 offers and 60 explicit-citation or action-alignment events across four high-level IDs, with no learning or persistent graph updates (Appendix A.3). Figure 1 traces recorded references and subsequent host actions as the task state evolves.

5 Conclusion

HPEM turns interaction experience into reusable processes and ordered workflows with persistent shared identities and node-level evidence. Its external memory interface supplies task-relevant guidance while preserving the host's reasoning and action control. Paired evaluations demonstrate strict-success gains across the evaluated WebShop 8B hosts and ALFWorld, including direct reuse of an 8B-acquired artifact by a 27B host. Matched-acquisition comparisons show higher strict completion than flat retrieval and higher reward with lower evaluation-time cost than trajectory retrieval. The online ablation shows improved frozen-evaluation success and efficiency with feedback-enabled acquisition, supporting cross-task process reuse.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript editing, including improving language, clarity, and presentation. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies experience reuse in interactive agents using WebShop, ALFWorld, and AndroidWorld. Experiments operate within benchmark environments. HPEM provides advisory process references, while the host agent retains control of every action. Freezing memory during evaluation separates experience acquisition from subsequent reuse. For deployment beyond these environments, stored experiences and retrieved guidance should remain subject to the application's privacy, access-control, and action-authorization requirements.

REPRODUCIBILITY STATEMENT

Section 3 specifies HPEM's process representation, composition rules, evidence updates, maturity criteria, and retrieval procedure. Section 4 describes the benchmarks, acquisition and evaluation settings, comparison conditions, and reported metrics. Appendix A provides implementation details, including adapter mappings, attribution rules, host configurations, frozen-memory controls, and statistical procedures. The matched-acquisition memory comparisons and online feedback ablation are distinguished explicitly, and earlier component studies retain their respective Full conditions. These descriptions identify the artifacts and experimental controls underlying the reported comparisons.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

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

Tao Feng, Chongrui Ye, Tianyang Luo, Jingjun Xu, Xueqiang Xu, Haozhen Zhang, Zhigang Hua, Yan Xie, Shuang Yang, Ge Liu, et al. Expgraph: Model-agnostic experience learning with graph-structured memory for llm agents. *arXiv preprint arXiv:2605.30712*, 2026a.

Yiyang Feng, Biddut Sarker Bijoy, Niranjan Balasubramanian, and Jiawei Zhou. Break it down, pass it on: Cross-task skill transfer in llm agents. *arXiv preprint arXiv:2608.20274*, 2026b.

Yu Hao, Jinxuan Cai, Qi Zhang, Yawen Li, Zhiqiang Zhang, Chuan Shi, and Cheng Yang. Hiskill: Empowering llm agents with hierarchical skill graphs. *arXiv preprint arXiv:2607.25853*, 2026.

Yuxin Jin, Siyuan Zhang, Hanchen Wang, Lu Qin, Ying Zhang, and Wenjie Zhang. Exg: Self-evolving agents with experience graphs. *arXiv preprint arXiv:2605.17721*, 2026.

Xiaoyuan Li, Moxin Li, Keqin Bao, Yubo Ma, Wenjie Wang, Dayiheng Liu, and Fuli Feng. Skillgraph: Skill-augmented reinforcement learning for agents via evolving skill graphs. *arXiv preprint arXiv:2605.12039*, 2026a.

Zhiyuan Li, Linyuan Gao, Xuechun Ding, Hongwei Chen, Yuan Wu, and Yi Chang. Caskg: Counterfactual-causal skill graphs for scalable agent skill retrieval. *arXiv preprint arXiv:2608.25500*, 2026b.

Dawei Liu, Zongxia Li, Hongyang Du, Xiyang Wu, Shihang Gui, Yongbei Kuang, and Lichao Sun. Graph-of-skills: Dependency-aware structural retrieval for massive agent skills. *arXiv preprint arXiv:2604.05333*, 2026.

Dang Nguyen, Viet Dac Lai, Seunghyun Yoon, Ryan A Rossi, Handong Zhao, Ruiyi Zhang, Puneet Mathur, Nedim Lipka, Yu Wang, Trung Bui, et al. Dynasaur: Large language agents beyond predefined actions. *arXiv preprint arXiv:2411.01747*, 2024.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Yuli Qiu, Yutong Li, Wei Su, Zeming Liu, Wanxiang Che, Heyan Huang, Haifeng Wang, and Yuang Guo. Lifemem: Enabling lifelong experience reuse for llm agents. *arXiv preprint arXiv:2609.12655*, 2026.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, volume 2025, pp. 406–441, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=rnKgbKmelt.

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. *arXiv preprint arXiv:2305.16291*, 2023.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Tianle Xia, Lingxiang Hu, Yiding Sun, Ming Xu, Lan Xu, Siying Wang, Wei Xu, and Jie Jiang. Grasp: Graph-structured skill compositions for llm agents. *arXiv preprint arXiv:2604.17870*, 2026.

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

Ruiyao Xu, Tiankai Yang, and Wei-Chieh Huang. Hyperskill: Self-evolving llm agents via hypergraph-structured skill memory. *arXiv preprint arXiv:2608.16114*, 2026a.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026b.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 37:113519–113544, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *Advances in Neural Information Processing Systems*, 38:12988–13018, 2026.

Jesse Zhang, Jiahui Zhang, Karl Pertsch, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun, and Joseph J Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. *arXiv preprint arXiv:2310.10021*, 2023.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025a.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025b.

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.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *International Conference on Learning Representations*, volume 2024, pp. 19036–19066, 2024.

A Implementation and Evaluation Details

A.1 Memory construction and retrieval

Adapter mappings. Benchmark-specific adapters provide semantic vocabularies, parsing rules, verification checks, and guidance templates. Acquisition determines which process variants and child combinations are stored and how their evidence accumulates. Concrete action arguments are abstracted, while context tags may retain task descriptors.

For WebShop, page types determine objectives and requested subgoals: `search` requests query formulation and candidate identification; `search_results` requests candidate identification; item pages request constraint verification, option configuration, and commitment. The task family is `product_purchase`, and context tags contain up to twelve unique non-stopword tokens from the instruction. Search-to-results and results-to-item transitions yield query and candidate observations. Changed item-page states yield verification or configuration observations; terminal commitment requires reward at least one. Text abstraction replaces concrete action arguments, product-ID patterns, and numeric values. Objective-specific templates provide expected success signals, applicability conditions, and failure boundaries.

For AndroidWorld, operation keywords in the task template and goal select an objective and planned subgoals. Previously recorded action intents are removed from the planned list, retaining the last planned subgoal when the remainder is empty. Action types and textual summaries map to a fixed intent vocabulary. Successful task traces supply leaf observations and ordered workflow children using unique intents in first-occurrence order. AndroidWorld uses task outcomes for downstream evidence. ALFWorld constructs its artifact offline from 24 successful training trajectories and retrieves references at AdaPlanner's planning and repair boundaries.

Matching and attribution. Leaf matching requires identical domain, objective, stage, task family, and normalized subgoals. Strategy-tag Jaccard similarity uses a threshold of 0.5, with similarity one for two empty sets. Workflow matching uses child objective–strategy profiles within the same intent scope and level, with a Jaccard threshold of 0.75. Both retain at most six non-retired variants per scope; reinforcement at capacity prioritizes similarity, reliability, and observation count. Workflow matching is order-insensitive, while storage and rendering preserve child order.

Each reference exposes either one direct root ID or the selected roots of a temporary composition. Declared IDs are intersected with the offered set. Rendering a descendant does not automatically assign it reference-use credit. Evidence is deduplicated using the most recent 64 task IDs per node and channel.

Native WebShop credits offered IDs returned by the host when local verification succeeds. ReAct credits offered roots on locally successful non-thought transitions; Reflexion uses the underlying ReAct execution mechanism. AndroidWorld uses explicit citations or deterministic action-intent compatibility for downstream attribution. At episode completion, credited offered nodes receive the task outcome; newly learned leaves and composed parents receive downstream success when the task succeeds. Multiple credited roots can receive the same episode outcome, with task-ID deduplication applied separately to each node.

Reference selection. Retrieval retains up to six trusted roots in the current domain, breaking ties by experience ID after rounding scores to six decimal places. The composer uses a final-score threshold of 0.18. Trusted roots satisfy $R(q_t, v) \geq 0.248$, so eligibility and ranking determine their inclusion in the shortlist.

A direct reference requires an exact-objective root covering the requested subgoals. Otherwise, temporary composition assigns a distinct qualifying root to each requested subgoal in order. Each selected root must cover its assigned subgoal and either match that subgoal as its objective or have all its subgoals within the request. An empty subgoal request falls back to the requested objective. A temporary reference requires complete coverage and at least two roots and remains episode-local. Eligibility is checked at selected roots; recursive rendering preserves stored child order.

A.2 HOST CONFIGURATIONS

WebShop. Native uses temperature zero, a 192-token action cap, and five recent actions in its JSON context. ReAct uses temperature zero, a fixed one-shot prompt, `think/search/click` grammar, one-line stopping, a 6,400-character prompt/history budget, and a 100-token action cap. Reflexion permits an initial attempt and three reflection retries, retains three reflections, and caps each reflection at 256 tokens. These runs allow 15 environment actions per attempt. Cost accounting includes all attempts and reflection calls.

The controlled memory and feedback studies use official ReAct with Qwen3-VL-8B-Instruct, 300 acquisition goals, and 500 evaluation goals. All evaluation arms use a 15-step limit, a 3,000-character reference budget, and retrieval top-$k = 6$. Flat process retrieval restricts the shared Full graph to trusted leaf candidates before ranking and top-k truncation. Trajectory retrieval uses successful trajectories from the same acquisition records.

The downstream-feedback ablation starts from an empty graph and disables task-level downstream evidence while retaining local process learning, local-use evidence, and composition. Full and the ablation are acquired online separately using the same acquisition goals, model, protocol, and budgets. Their interaction trajectories can therefore differ as their memories evolve.

ALFWorld. AdaPlanner uses temperature zero, a 1,400-token completion cap, a 50-action budget, and its six-attempt checkpoint-repair/reset policy. Thinking is enabled for 8B and disabled for 27B. Both models use the same 8B-acquired artifact. Retrieval occurs at planning and repair boundaries, with at most eight offers per episode. Calls, tokens, and wall time are recorded by the same transport.

AndroidWorld. M3A uses temperature zero, a 16,384-token context, at most six offers per episode, and an action budget of $\mathrm{int}(10 \times \text{task complexity})$.

A.3 EVALUATION, STATISTICS, AND ARTIFACT AUDITS

Paired statistical procedures. Confidence intervals use paired bootstrap resampling over goals for WebShop, tasks for ALFWorld, and task classes for AndroidWorld. Primary WebShop comparisons use 10,000 resamples; ALFWorld, 27B ExpeL, and the controlled ReAct memory and feedback studies use 5,000. Percentile endpoints are 2.5 and 97.5. Each resampled evaluation unit retains both arms of its comparison. Intervals describe task-level variation conditional on the evaluated artifacts and configurations. The 27B ExpeL intervals are unadjusted across hosts and versions. Infrastructure errors trigger symmetric pair exclusion; the two controlled ReAct studies report zero infrastructure errors.

For the controlled ReAct studies, Full-only/comparator-only success counts are 64/37 against Flat, 55/41 against Trajectory, and 80/36 against the no-feedback variant. Two-sided exact McNemar tests use the binomial distribution with probability $1/2$, conditional on the number of discordant pairs. The corresponding p-values are 0.00934, 0.184, and 5.40×10^{-5}. Holm correction over the two memory-baseline success comparisons gives an adjusted $p = 0.0187$ for Full versus Flat; the feedback test is reported separately. Bootstrap intervals are pointwise.

Full-minus-comparator step differences and paired 95% intervals are $+1.15$ $[0.82, 1.51]$ for Flat, -0.56 $[-0.93, -0.18]$ for Trajectory, and -1.53 $[-1.97, -1.11]$ for the no-feedback variant. Differences are computed before rounding and may therefore differ from differences between the displayed means.

Protocol-specific component comparisons. Table 7 reports component comparisons under their respective acquisition protocols. The original comparison uses a Full condition with 29.0% success; the rebuilt-memory comparisons use a Full condition with 22.6% success. These conditions remain separate from the 30.0% Full condition in the controlled study. Each contrast is interpreted within its own protocol.

Frozen-memory checks. Frozen WebShop and AndroidWorld runs disable process observation, downstream updates, persistent workflow creation or reinforcement, and memory saving. Read-only retrieval and temporary reference composition remain enabled. The controlled ReAct studies report

Table 7: WebShop ReAct component comparisons on 500 paired goals. Success is in percent; Δ is Full minus ablation in pp. Each comparison uses its protocol-specific Full condition.

Removed capability	Full	Ablation	Δ [95% CI]
Original acquisition-matched protocol			
Downstream feedback	29.0	20.2	**+8.8 [4.6, 12.8]**
Rebuilt-memory protocol			
Shared child identity	22.6	16.6	**+6.0 [3.2, 8.8]**
Cross-workflow composition	22.6	20.6	+2.0 [−0.8, 4.8]

Table 8: Frozen memory audits for the primary host comparisons. High-level counts include trusted composed nodes; depth is the number of hierarchy levels. Events are declared uses for Native and alignment records for ReAct, Reflexion, and AndroidWorld.

Host	Nodes	Trusted	High-level	Depth	Offers	Events
WebShop Native	15	10	4	3	1,754	228
WebShop ReAct	9	9	5	4	4,971	1,574
WebShop Reflexion	11	9	5	4	19,990	3,048
AndroidWorld M3A	43	17	9	3	95	60

no learning or graph updates during evaluation. For the controlled WebShop comparisons, artifact hashes confirm that Full and Flat use identical frozen memory states. All four arms share the same 500 evaluation goals, and recorded graph versions remain unchanged before and after every episode. ALFWorld records a consistent memory digest across each full evaluation; a separate 27B canary verifies matching before/after full-file hashes.

In the controlled feedback study, Full and the no-feedback variant contain nine versus seven nodes and twelve versus seven edges. They have nine versus two trusted nodes, five versus two trusted high-level nodes, and four versus two hierarchy levels. Downstream evidence events number 1,029 versus zero. Memory is offered in 500 versus 246 evaluation episodes. The no-feedback variant retains 592 learning events and five composition events. These event counts include creation and reinforcement.

Controlled-artifact evidence audit. The controlled Full artifact contains nine trusted nodes: four leaves and five composed workflows, reaching level four. Independent recomputation of the runtime maturity rules agrees with every stored label. Seven nodes use the local-evidence branch, including all five composed nodes; two leaves use the downstream branch. Replacing only ρ_v with $\rho_v^{\mathrm{pool}} = (S_v^{\mathrm{L}} + S_v^{\mathrm{D}} + 1)/(N_v^{\mathrm{L}} + N_v^{\mathrm{D}} + 2)$, while retaining A_v, B_v, O_v, and the maturity thresholds, changes none of the nine labels in the frozen snapshot. Three composed roots are offered in 290, 165, and 500 evaluation episodes, respectively, documenting that stored workflows enter the reference channel during frozen reuse.

Reference-length accounting. Under the same 3,000-character budget, Full and Flat average 884.1 and 1,454.0 characters across 4,886 and 4,309 reference-applied model-action records, respectively. Lengths measure rendered bodies after budget enforcement, excluding prompt wrappers, with no missing records. Across the same 500 goals, cumulative reference length averages 8,639.8 versus 12,530.6 characters per episode. Full's higher strict success thus accompanies shorter references per record and per episode.

Structural audits and recorded interaction. Table 8 summarizes the frozen artifacts. AndroidWorld acquisition records 80 offers, 37 aligned events, 164 learning events, and 45 composition events. Frozen evaluation records 95 offers and 60 aligned events spanning four high-level IDs, with zero learning or composition events. Its 43 nodes and 17 trusted nodes remain unchanged between phases. Recursive traversal from its nine trusted composed roots yields 28 root–descendant pairs, all with trusted descendants in the audited snapshot.

Figure 1 reconstructs a retained frozen `MarkorDeleteNote` episode and its corresponding memory snapshot. Recipe and expense workflows reference the same stored child IDs. At host decision steps $t = 0, 1, 4$, the recorded prompts offer recipe, recipe, and expense references, respectively. The subsequent host actions are opening Markor, long-pressing the target note, and declaring completion. At $t = 4$, the target is already absent. The host retains discretion over each action. Markor also occurs during acquisition; the held-out case is a new task instance.