DUAL-MEM: CROSS-TASK MEMORY FORMATION AND WITHIN-TASK ACTIVE MEMORY CONTROL FOR TOOL-USING LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using large language model (LLM) agents can encode past interaction experiences into procedural memory. However, memory retrieval alone does not determine how retrieved memory should be adapted during execution. When execution stalls, the agent must decide whether to revise the current strategy or replace the retrieved procedure. We propose DUAL-MEM, a dual lifecycle framework that separates Strategy adaptation within a task from procedural memory evolution across tasks. During execution, an event-driven Controller decides whether to regenerate the Strategy while preserving the current Card or to retrieve a different Card when the procedure no longer provides suitable guidance. After each task, observed usage outcomes update Card history, while successful unmatched trajectories can contribute new procedural memory for later tasks. Across four interactive benchmarks, DUAL-MEM improves average success rate from 46.30% to 49.30% over the strongest baseline. On ALFWorld, success improves by 2.98%–3.73% across three different model backbones, while average online input tokens are reduced by 4.10% to 4.50%. Over a 120-task continual stream, fixed probe success increases from 68.91% to 73.88%. Execution traces indicate a clear timescale separation between the two levels. Active Cards have an RMST@40 2.27 times that of Strategies, suggesting that Strategy revision often occurs without replacing the underlying procedural memory.

1 INTRODUCTION

Large language model (LLM) agents have shown strong performance on long-horizon tasks that require planning and tool use in changing environments (Yao et al., 2023; Ahn et al., 2022; Wang et al., 2023a). Unlike single-turn generation, these tasks require a sequence of decisions grounded in intermediate observations. Execution can fail as the environment changes, making an initially reasonable plan ineffective. Reliable agents must therefore adapt their current behavior without repeatedly exploring problems that prior experience has already solved.

Existing work improves this ability along two related directions. Reasoning-and-acting agents incorporate environmental feedback into successive decisions, while planning-based methods revise plans during execution (Huang et al., 2022; Wang et al., 2023b; Sun et al., 2023). Experience-driven agents extend adaptation across tasks by reusing information from previous interactions. Prior experience has been stored as linguistic feedback or as executable procedural structures such as skills and workflows (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2023a; 2025; Fang et al., 2026). Procedural memory is particularly suitable for long-horizon tasks because successful interaction can be compressed into reusable procedures that directly guide later execution.

Reusing procedural memory nevertheless creates a distinct ambiguity during execution. A retrieved procedure may remain applicable even after the task-specific Strategy derived from it has become stale. For example, a procedure such as "locate the object, transform it, then place it" may still be valid when the current Strategy keeps searching an unproductive location. In this case, execution should adapt without discarding useful procedural knowledge. A different failure may indicate that the procedure itself no longer fits the current state. Across tasks, the memory system must also decide which experiences deserve persistent storage and how previous usage should affect later reuse (Cao

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

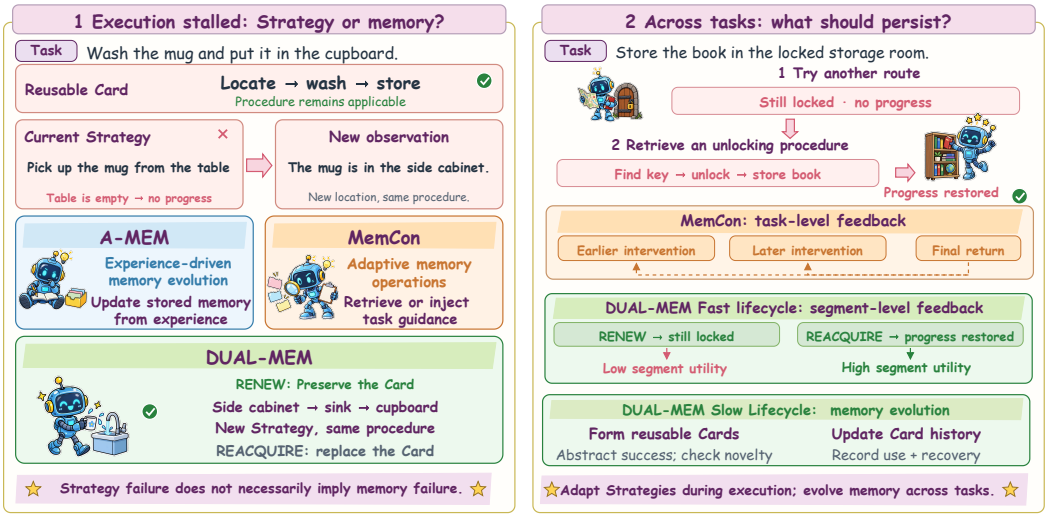


Figure 1: Motivation for DUAL-MEM. The left panel shows that an execution stall may reflect stale task-specific guidance even when the reusable procedure remains applicable, motivating separate RENEW and REACQUIRE interventions. The right panel shows that task-level feedback is coarse when several interventions occur within one trajectory, while segment-level outcomes provide more localized evidence for future memory use.

et al., 2026; Ye et al., 2026; Jiang et al., 2026). These cases expose a coordination problem between local execution adaptation and persistent memory evolution.

We introduce DUAL-MEM, a dual-lifecycle framework that represents procedural guidance at two levels. A Card is the persistent procedural-memory unit and pairs an applicability description with a reusable procedure abstracted from successful interaction. When activated for a task, the Card is instantiated as a Strategy that provides task-specific guidance under the current execution context. This separation allows the procedure to persist when only its current realization needs to change.

The Slow Lifecycle governs how Cards evolve across tasks and records evidence from previous use. The Fast Lifecycle operates on the active Card and Strategy during execution. When progress stalls, the Controller chooses between two intervention levels. RENEW preserves the Card and regenerates its Strategy from recent execution evidence. REACQUIRE replaces the Card when its procedure no longer provides suitable guidance. The execution segment following each intervention provides localized utility, which is written back to Card history and later informs control.

We evaluate DUAL-MEM through cross-benchmark experiments and controlled lifecycle analyses. Across four interactive environments, DUAL-MEM improves average success from 46.30% to 49.30% over the strongest baseline. With a frozen Memory Bank, contextual control reaches 68.91% success, compared with 62.94% under fixed RENEW and 61.44% under fixed REACQUIRE. Continual evaluation further raises fixed-probe success from 68.91% to 73.88% when Bank evolution is combined with accumulated usage evidence. Execution traces show that Cards remain active 2.27 times longer than Strategies, consistent with their intended roles at different temporal scales.

Our contributions are threefold. **(1) Dual-lifecycle formulation.** We formulate procedural-memory management at two coupled temporal scales. The Slow Lifecycle evolves reusable memory across tasks, while the Fast Lifecycle controls how active memory guides the current task. **(2) Hierarchical active-memory control.** We separate persistent procedural Cards from task-specific Strategies. Execution-time control chooses between RENEW for Strategy revision and REACQUIRE for Card replacement. **(3) Closed-loop persistent-memory update.** We connect intervention outcomes with persistent Card history so that execution experience can inform future control and selective Card formation.

2 Related Work

2.1 LLM Agents for Interactive Decision Making

Large language models have been developed as agents that pursue goals through sequential interaction with external environments. ReAct interleaves reasoning with actions so that intermediate observations can affect later decisions (Yao et al., 2023). In embodied settings, SayCan grounds language-level decisions in executable skills, while Inner Monologue incorporates environment feedback into planning (Ahn et al., 2022; Huang et al., 2022). DEPS and AdaPlanner further revise long-horizon behavior as execution feedback becomes available (Wang et al., 2023b; Sun et al., 2023). These methods primarily adapt within a task, leaving open how experience from completed interactions should persist and guide later tasks.

2.2 Memory-Augmented LLM Agents

Memory-augmented agents carry experience across tasks in increasingly structured forms. Reflexion stores linguistic feedback, ExpeL distills transferable experience, and Voyager builds a reusable skill library from successful interaction (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2023a). AWM extracts reusable workflows, while MemP constructs and refines procedural experience for later execution (Wang et al., 2025; Fang et al., 2026). ReMe maintains an evolving procedural-memory pool through cross-task refinement (Cao et al., 2026).

Recent work also treats memory use itself as a control problem. Memory-R1 learns memory modification policies through interaction (Yan et al., 2026), MemCon makes memory access responsive to the current execution state (Jiang et al., 2026), and CoEvo-Mem jointly develops retrieval behavior and the memory bank over continual task streams (Ye et al., 2026). These methods make persistent memory increasingly adaptive, while task-local execution adaptation is usually handled separately. DUAL-MEM addresses this interface by using execution feedback to decide whether to revise the current Strategy or replace the procedural Card, linking within-task control to cross-task memory formation.

3 Methodology

3.1 Dual-Lifecycle Framework

DUAL-MEM separates procedural-memory management by update timescale. During a task, the Fast Lifecycle controls how active procedural knowledge guides execution while the persistent memory state remains fixed. After the task terminates, the Slow Lifecycle updates the information carried across tasks. The persistent procedural unit is a Card, and each active Card is instantiated into a task-specific Strategy.

Consider a sequence of interactive tasks $q_0, \ldots, q_{N-1}$. For task q_n, the agent interacts with the environment and records the execution trajectory

$$\tau_n = (o_0, a_0, o_1, a_1, \ldots, a_{T_n-1}, o_{T_n}), \tag{1}$$

where o_t and a_t denote the observation and action at step t. Before task q_n begins, the system carries a Memory Bank $\mathcal{B}_n$ and a Usage State $\mathcal{U}_n$. The Memory Bank stores reusable procedural knowledge. The Usage State records the reliability u_i and usage support v_i associated with each Card. Both remain fixed during task q_n. Each procedural Card is represented as

$$m_i = (\omega_i, c_i), \qquad m_i \in \mathcal{B}_n, \tag{2}$$

where ω_i describes the conditions under which the Card is applicable and c_i stores its reusable procedure. Retrieval operates on ω_i, while c_i provides the procedural content used during execution.

At task start, the agent retrieves a Card from the task goal and initial observation. We denote the Strategy Generator by F and the Actor by A. The retrieved procedure is instantiated into a Strategy σ_j, after which the Actor produces environment actions according to

$$\sigma_j = F(q_n, o_t, \mathcal{H}_t, c_i), \qquad a_t = A(q_n, o_t, \mathcal{H}_t, \sigma_j), \tag{3}$$

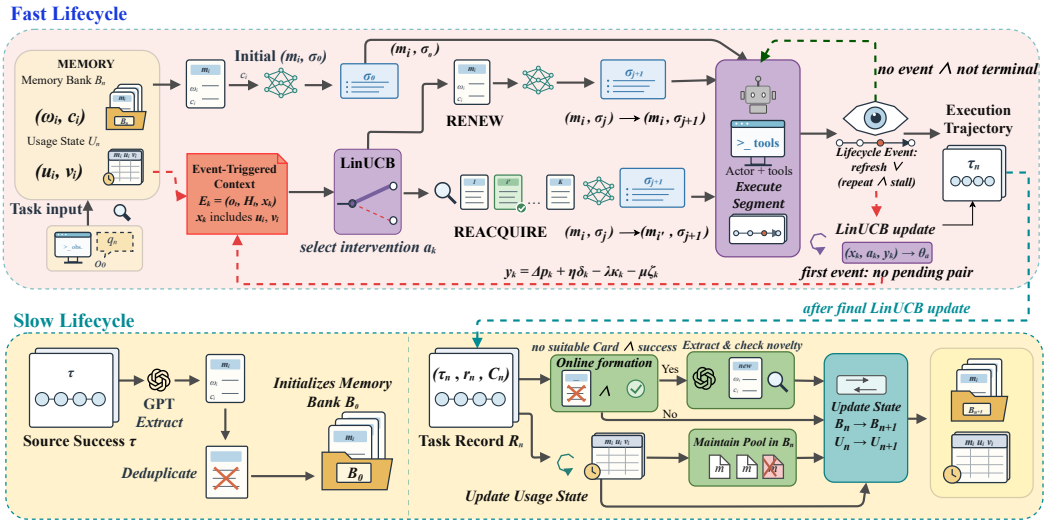


Figure 2: Overview of DUAL-MEM. During task q_n, the Fast Lifecycle controls the active Card and its task-specific Strategy while $\mathcal{B}_n$ and $\mathcal{U}_n$ remain fixed. At task completion, the Task Record R_n is passed to the Slow Lifecycle, which updates the persistent state for task q_{n+1}.

where $\mathcal{H}_t$ is the recent interaction history. The Strategy may remain active over several environment steps, whereas the Actor produces a new action at every step. We use the same backbone LLM for F and A with separate prompts. A Card may remain active across successive Strategies, allowing task-specific guidance to change without immediately replacing the underlying procedure.

At task termination, the execution trajectory, task outcome, and Card-use evidence are collected into a Task Record R_n. The Task Record provides the interface between the two lifecycles and is processed before task q_{n+1} begins. Figure 2 summarizes the resulting closed loop.

3.2 Slow Lifecycle for Cross-Task Memory Evolution

Before the target task stream begins, the initial Memory Bank $\mathcal{B}_0$ is constructed from successful trajectories collected on separate source tasks. Each trajectory is summarized into a Card whose procedure captures reusable solution structure and whose applicability description supports later retrieval. During the target stream, a successful trajectory may contribute a new Card when the existing Bank does not provide suitable procedural guidance.

A candidate Card is admitted only when its applicability description is sufficiently different from those already stored. For a candidate description $\tilde{\omega}$, admission requires

$$\max_{m_i \in \mathcal{B}_n} \cos\big(\mathrm{emb}(\tilde{\omega}), \mathrm{emb}(\omega_i)\big) < \epsilon.$$

The threshold ϵ controls memory admission. Smaller values impose stricter deduplication, while larger values permit more similar Cards to enter the Bank. We admit at most one new Card from each completed task.

3.3 Fast Lifecycle for Within-Task Memory Control

The Slow Lifecycle also updates the Usage State from the Card-use evidence in R_n. For each activated Card, u_i summarizes its historical reliability and v_i reflects the amount of supporting usage evidence. These quantities are supplied to the Fast Controller in later tasks, while semantic retrieval continues to depend on ω_i. The updated $\mathcal{B}_{n+1}$ and $\mathcal{U}_{n+1}$ are then carried to task q_{n+1}. Appendix B gives the operational definition of Card suitability and specifies how usage evidence updates the persistent state. It also describes maintenance of the active retrieval pool.

Once a Card and Strategy are active, the Actor continues normal interaction until the Watcher detects a lifecycle event. The Watcher responds to an explicit refresh request or to sustained execution

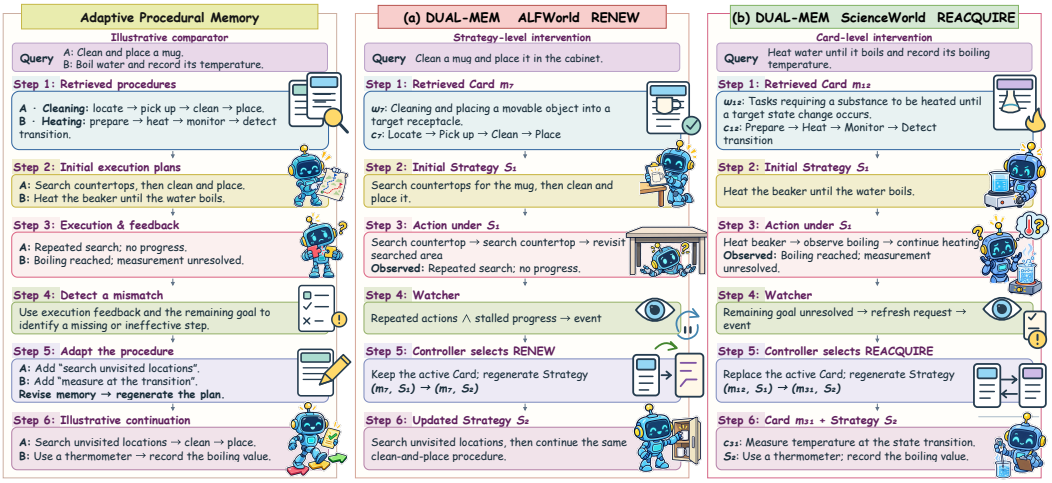


Figure 3: Examples of Fast Lifecycle intervention. The left panel gives an illustrative example of generic procedure adaptation and is not an experimental baseline. In the center ALFWorld example, RENEW preserves the active Card and revises its Strategy. In the right ScienceWorld example, REACQUIRE replaces the Card and produces new task-specific guidance.

stagnation. We use a stall threshold h to determine when lack of progress becomes actionable. A stall event is triggered when repeated behavior persists without observable progress for h consecutive steps. The Controller is invoked only at such events, so ordinary execution remains outside its action space. At event k, the Watcher records

$$E_k = (o_{t_k}, \mathcal{H}_{t_k}, \chi_k), \tag{4}$$

where o_{t_k} is the current observation, $\mathcal{H}_{t_k}$ is the recent interaction history, and χ_k identifies the trigger. The Controller selects one of two interventions,

$$\mathcal{A}_{\mathrm{F}} = \{\textsc{Renew}, \textsc{Reacquire}\}.$$

The RENEW operation preserves the active Card and regenerates its Strategy using the latest execution evidence,

$$(m_i, \sigma_j) \xrightarrow{\textsc{Renew}} \big(m_i, F(q_n, o_{t_k}, \mathcal{H}_{t_k}, c_i)\big). \tag{5}$$

This changes task-specific guidance without discarding the underlying procedural knowledge.

The REACQUIRE operation replaces the active Card. A state-aware query is first constructed from the current task and event evidence,

$$z_k = Q(q_n, E_k), \qquad i' = \arg\max_{i \in \mathcal{I}_k} \cos\big(\mathrm{emb}(z_k), \mathrm{emb}(\omega_i)\big), \qquad \sigma_{j+1} = F(q_n, o_{t_k}, \mathcal{H}_{t_k}, c_{i'}), \tag{6}$$

where $\mathcal{I}_k \subseteq \mathcal{B}_n$ contains the Cards that remain available during the current task. The query reflects the current execution state rather than the initial task state. Cards already abandoned during the task are excluded from subsequent retrieval. If no suitable Card remains, execution continues without an active Card.

At each lifecycle event, the Controller constructs the context

$$x_k = [1, b_k, d_k, \ell_k^{\sigma}, \ell_k^{m}, r_k, \rho_k, u_i, v_i]^{\top}. \tag{7}$$

The feature b_k indicates repeated-action behavior, and d_k measures the time since the last observed progress. The variables ℓ_k^{σ} and ℓ_k^{m} denote the ages of the active Strategy and Card. The value r_k records previous RENEW operations under the current Card. The applicability feature ρ_k measures similarity between the current execution context and the active Card. The long-term features u_i and v_i are read from $\mathcal{U}_n$. Their detailed computation is given in Appendix B.

We use LinUCB (Li et al., 2010) to select between the two interventions,

$$a_k = \arg\max_{a \in \mathcal{A}_{\mathrm{F}}} \left[x_k^{\top} \hat{\theta}_a + \alpha \sqrt{x_k^{\top} A_a^{-1} x_k} \right]. \tag{8}$$

The coefficient α controls the strength of uncertainty-driven exploration. A larger value assigns greater weight to interventions whose utility estimates remain uncertain.

After the selected intervention, execution continues until the next lifecycle event or task termination. This interval defines an execution segment with utility

$$y_k = \Delta p_k + \eta \delta_k - \lambda \kappa_k - \mu \zeta_k, \tag{9}$$

where Δp_k measures progress during the resulting segment and δ_k accounts for task completion. The remaining terms penalize intervention cost and continued stagnation. After the segment ends, the observed utility updates the selected LinUCB arm.

At the first lifecycle event, no update is pending because the initial Card and Strategy precede the Controller's first decision. At task termination, the final pending LinUCB update is completed before the Task Record R_n is formed. The Slow Lifecycle then processes R_n to obtain $\mathcal{B}_{n+1}$ and $\mathcal{U}_{n+1}$ for the next task.

4 Experiments

4.1 Benchmarks and Metrics

We evaluate DUAL-MEM on four interactive benchmarks. ALFWorld measures household interaction, AppWorld evaluates API-based tasks, ScienceWorld requires multi-step scientific experimentation, and τ-bench tests tool use under user interaction (Shridhar et al., 2021; Trivedi et al., 2024; Wang et al., 2022; Yao et al., 2024). We additionally include TriviaQA as a shorter-horizon knowledge benchmark and GAIA as a general-assistant setting with multi-step tasks (Joshi et al., 2017; Mialon et al., 2023). Each benchmark is evaluated with its official primary metric. Interactive Avg. is the macro average over the four interactive environments, while QA Avg. averages TriviaQA and GAIA. The evaluation splits and instance counts are reported in Appendix A, Table 3.

4.2 Evaluation Protocols

Cross-benchmark evaluation. Each benchmark constructs its initial Memory Bank from the corresponding training split. Methods compared within the same evaluation run share the execution backbone, task order, tool interface, and evaluator. Dynamic baselines retain their native memory updates during evaluation. No future evaluation task is used to initialize the Bank. We conduct three independent evaluation runs. For stateful memory methods, the runs use different task-order seeds to capture order-dependent memory dynamics. All generation calls use temperature-zero decoding; hosted-model inference, however, is not assumed to be bitwise deterministic, and therefore stateless methods may also exhibit small run-to-run variation. We report the mean and standard deviation across the three runs.

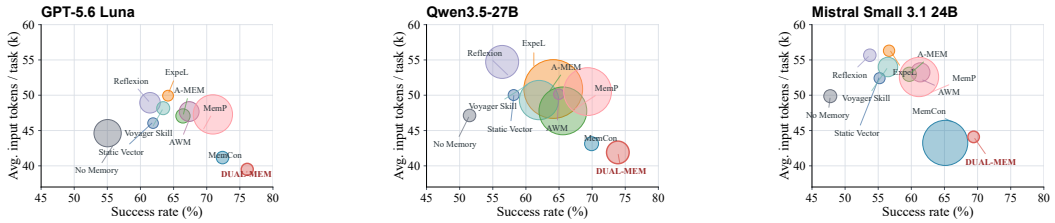
Controlled lifecycle analysis. The controlled studies use ALFWorld with Qwen3.5-27B. All variants start from the same 60-Card Bank and are evaluated on the same 134 held-out tasks. The Fast Lifecycle study freezes the Bank, retrieval index, Watcher, and Usage State so that only the intervention policy changes. Coupled Refresh removes the explicit RENEW/REACQUIRE distinction by rerunning state-aware Card retrieval over the full eligible pool, including the active Card, and regenerating the Strategy at every lifecycle event. The Slow Lifecycle study processes a shared 120-task training stream before the resulting memory state is frozen for evaluation on the same held-out probe. Unless varied explicitly, we use $h = 4$, $\epsilon = 0.85$, and $\alpha = 0.50$ with a maximum episode length of 50 environment steps.

4.3 Implementation and Baselines

The main six-benchmark evaluation uses GPT-5.6 Luna (OpenAI, 2026a). Controlled lifecycle analyses use Qwen3.5-27B (Qwen Team, 2026), and the ALFWorld efficiency study additionally evaluates Mistral-Small-3.1-24B-Instruct (Mistral AI, 2025). We compare against non-memory and static-retrieval baselines and representative experience- and memory-based agents: Reflexion, ExpeL, Voyager-style, MemoryBank, A-MEM, AWM, MemP, MemCon, and ReMe (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2023a; Zhong et al., 2024; Xu et al., 2025; Wang et al., 2025; Fang et al.,

Table 1: Overall performance across six benchmarks with GPT-5.6 Luna. Values are mean ± standard deviation over three independent evaluation runs. Stateful methods use a different task-order seed in each run, while all methods within a run share the same task order. Interactive Avg. is the macro average over ALFWorld, AppWorld, ScienceWorld, and τ-bench. QA Avg. averages TriviaQA and GAIA.

	Interactive Environments					QA / Assistant Tasks		
Method	**ALFWorld**	**AppWorld**	**SciWorld**	**τ-bench**	**Avg.**	**TriviaQA**	**GAIA**	**Avg.**
No Memory	43.28(±1.97)	27.78(±1.50)	29.63(±1.61)	21.14(±1.27)	30.46(±1.51)	72.10(±0.70)	24.65(±1.40)	48.37(±0.97)
Static Vector	55.72(±1.55)	33.33(±1.19)	36.54(±1.40)	25.81(±0.93)	37.85(±1.26)	72.80(±0.60)	27.27(±1.21)	50.04(±0.89)
Reflexion	53.98(±1.72)	31.94(±1.37)	35.06(±1.50)	24.80(±1.27)	36.45(±1.36)	73.40(±0.60)	28.08(±1.26)	50.74(±0.93)
ExpeL	59.70(±1.49)	35.71(±1.19)	38.64(±1.30)	27.44(±1.06)	40.37(±1.23)	73.70(±0.50)	29.09(±1.05)	51.40(±0.74)
Voyager-style	57.71(±1.55)	34.33(±1.24)	37.78(±1.34)	26.83(±1.06)	39.16(±1.23)	73.10(±0.60)	28.69(±1.26)	50.89(±0.88)
MemoryBank	58.96(±1.49)	34.92(±1.24)	37.16(±1.30)	26.22(±1.06)	39.31(±1.22)	73.80(±0.50)	28.28(±0.93)	51.04(±0.71)
A-MEM	62.19(±1.14)	37.70(±1.24)	40.37(±1.28)	29.07(±0.93)	42.33(±1.11)	74.10(±0.50)	30.10(±0.93)	52.10(±0.71)
AWM	63.68(±1.14)	38.29(±1.24)	41.11(±1.11)	29.88(±1.06)	43.24(±1.10)	74.20(±0.50)	30.91(±1.05)	52.55(±0.77)
MemP	66.17(±1.14)	40.28(±0.91)	41.98(±0.93)	31.30(±0.70)	44.93(±0.90)	74.40(±0.40)	31.92(±0.93)	53.16(±0.66)
MemCon	68.41(±1.14)	41.27(±0.91)	43.21(±0.93)	32.32(±0.61)	46.30(±0.89)	74.50(±0.40)	32.93(±0.70)	53.71(±0.54)
DUAL-MEM	**72.14**(±0.86)	**44.05**(±0.60)	**46.05**(±0.93)	**34.96**(±0.70)	**49.30**(±0.72)	**75.20**(±0.40)	**35.56**(±0.70)	**55.38**(±0.52)
Gain vs. MemCon	**+3.73**	**+2.78**	**+2.84**	**+2.64**	**+3.00**	**+0.70**	**+2.63**	**+1.67**



(a) (b) (c)

Figure 4: Success–cost trade-offs on the 134-task ALFWorld suite across three execution backbones. Each marker summarizes three task-order seeds; token counts are compared only within a backbone.

2026; Jiang et al., 2026; Cao et al., 2026). MemCon is the closest adaptive memory-control baseline. All memory-based methods are constructed from the same benchmark-specific source pool; each method retains its native memory representation, construction procedures and update mechanism, and uses the default hyperparameters reported in the original work. Reproduction details are provided in Appendix A.

Efficiency results use online input tokens consumed during task execution. The count includes Actor calls, Strategy generation, and retrieval or reacquisition prompts. Initial Bank construction and post-task Bank evolution are excluded from this metric.

5 Results and Analysis

5.1 Overall Performance and Efficiency

Cross-benchmark performance. Table 1 reports the main six-benchmark results with GPT-5.6 Luna. Across the four interactive environments, DUAL-MEM improves the macro average from 46.30% to 49.30% over MemCon, the strongest full-suite baseline. On AppWorld, it also outperforms the reproduced ReMe baseline (44.05% vs. 41.87%; Table 4 in Appendix A). The improvement is smaller on TriviaQA, while GAIA retains a 2.63-point gain. The larger effect in interactive environments is consistent with the role of lifecycle control, since these tasks repeatedly expose the agent to execution states in which procedural guidance may need revision.

Inference efficiency. Figure 4 compares success and online input-token cost on the same 134-task ALFWorld suite under three execution backbones. Relative to MemCon, DUAL-MEM improves

Table 2: Controlled lifecycle ablations on ALFWorld. Panel (a) freezes the persistent memory state and varies the Fast Lifecycle intervention policy. Panel (b) fixes the Fast Lifecycle and varies cross-task Bank evolution and usage evidence. The LinUCB row in panel (a) is repeated as the Frozen Persistent State anchor in panel (b).

(a) Fast Lifecycle control

Variant	Success (%) ↑	Steps ↓	Input tokens/task ↓
No Controller	57.71$_{(\pm 1.14)}$	29.10$_{(\pm 0.80)}$	58.40$_{(\pm 1.90k)}$
Always Renew	62.94$_{(\pm 1.14)}$	25.40$_{(\pm 0.70)}$	59.70$_{(\pm 1.80k)}$
Always Reacquire	61.44$_{(\pm 1.14)}$	26.30$_{(\pm 0.80)}$	65.80$_{(\pm 2.10k)}$
Coupled Refresh	64.93$_{(\pm 1.49)}$	25.10$_{(\pm 0.40)}$	59.80$_{(\pm 1.20k)}$
Non-contextual UCB	65.17$_{(\pm 1.14)}$	24.30$_{(\pm 0.60)}$	56.90$_{(\pm 1.70k)}$
LinTS	66.92$_{(\pm 1.14)}$	23.60$_{(\pm 0.70)}$	54.80$_{(\pm 1.60k)}$
LinUCB (DUAL-MEM)	**68.91**$_{(\pm 1.55)}$	**23.00**$_{(\pm 0.40)}$	**52.60**$_{(\pm 1.50k)}$

(b) Slow Lifecycle evolution

Variant	Bank Evolution	Usage History	Success (%) ↑	Steps ↓	Input tokens ↓
Frozen Persistent State	×	×	68.91$_{(\pm 1.55)}$	23.00$_{(\pm 0.40)}$	52.60$_{(\pm 1.50k)}$
History Only	×	✓	70.15$_{(\pm 1.49)}$	22.50$_{(\pm 0.40)}$	44.13$_{(\pm 0.40k)}$
Bank Evolution Only	✓	×	71.64$_{(\pm 1.49)}$	22.10$_{(\pm 0.40)}$	43.73$_{(\pm 0.31k)}$
Full DUAL-MEM	✓	✓	**73.88**$_{(\pm 1.49)}$	**20.90**$_{(\pm 0.40)}$	**41.90**$_{(\pm 0.26k)}$

success by 3.73 points while reducing average input tokens by 4.10% with GPT-5.6 Luna. The corresponding changes are 3.73 points and 4.50% with Qwen3.5-27B. With Mistral-Small-3.1-24B-Instruct, success increases by 2.98 points while input tokens decrease by 4.40%. The same success-efficiency direction holds across all three backbones.

5.2 Lifecycle Ablations

Fast Lifecycle control. Table 2(a) freezes the persistent memory state and varies only the intervention policy. Both fixed interventions improve over No Controller, while Coupled Refresh further benefits from state-aware guidance updates. However, Coupled Refresh reaches 64.93% success with 59.80k input tokens per task, compared with 68.91% and 52.60k for contextual LinUCB. This gap supports separating Strategy revision from Card replacement: jointly refreshing both levels at every event can trigger unnecessary memory retrieval when the underlying procedure remains applicable.

Slow Lifecycle evolution. Table 2(b) fixes the Fast Lifecycle and varies the persistent memory update. History Only raises success from 68.91% to 70.15%, while Bank Evolution Only reaches 71.64%. Combining both reaches 73.88% and reduces average input tokens from 52.60k to 41.90k. Bank evolution broadens the procedures available for later tasks. Usage history instead changes the evidence available to the Controller when an activated Card is evaluated. The two gains therefore arise from different parts of the lifecycle.

5.3 Continual Evolution and Card–Strategy Dynamics

Continual memory formation. Figure 5(a) tracks the held-out probe during the 120-task continual stream. Fixed-probe success rises from 68.91% to 73.88% as the mean Bank grows from 60.00 to 76.67 Cards. This trajectory complements the endpoint ablation in Table 2(b) by showing how the benefit of memory evolution accumulates as additional tasks are observed.

Card–Strategy persistence. Figures 5(b) and 5(c) describe the persistence of the two active states during execution. Across 402 ALFWorld episodes, Card RMST@40 reaches 20.27 ± 0.65 environment steps, compared with 8.93 ± 0.21 for Strategies. A total of 53.10% of active Card spans contain at least two successive Strategies. Strategy revision therefore occurs frequently while the same Card remains active. Detailed lifetime statistics are reported in Appendix C, Tables 7–9.

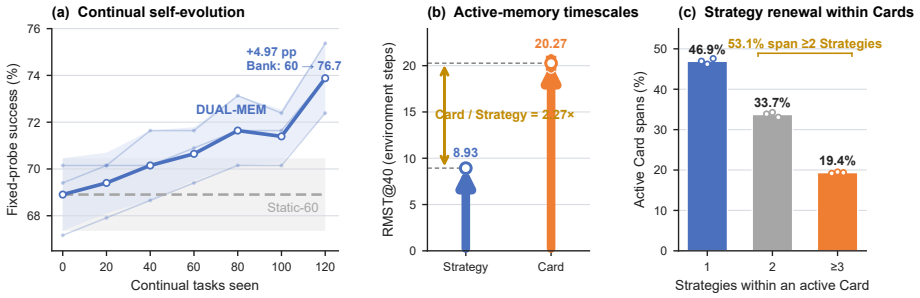


Figure 5: Continual evolution and Card–Strategy execution dynamics: (a) fixed-probe performance, (b) Strategy and Card RMST@40, and (c) successive Strategies per Card. Error summaries use three task-order seeds.

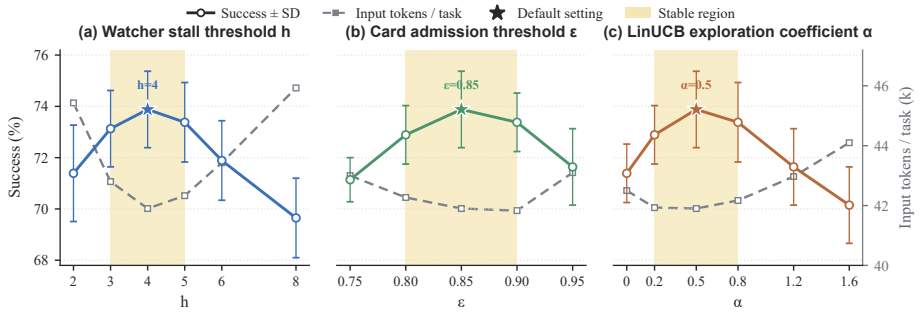


Figure 6: Hyperparameter sensitivity on ALFWorld with Qwen3.5-27B. Stars mark defaults; shading marks settings within one success point.

Interpretation. The lifetime difference describes execution under the control structure defined by DUAL-MEM. Because RENEW ends a Strategy span by construction, while a Card span ends only after REACQUIRE or task termination, these measurements should be read as execution dynamics rather than independent validation of the Card–Strategy hierarchy.

5.4 HYPERPARAMETER SENSITIVITY

Stable operating regions. Figure 6 varies one parameter at a time under the continual evaluation protocol. Mean success remains within one percentage point of the default for $h \in [3, 5]$. The corresponding stable region for Card admission is $\epsilon \in [0.80, 0.90]$. For LinUCB, $\alpha \in [0.20, 0.80]$ gives the same level of stability. The default configuration therefore lies inside a nontrivial stable region for each parameter.

Behavior outside the stable region. A small stall threshold triggers intervention more frequently, whereas a large value delays recovery. Increasing ϵ admits more Cards, but the larger Bank does not yield sustained gains at the upper end of the sweep. Large α values place more weight on uncertain interventions and increase online inference cost. The complete sweeps and event statistics are reported in Appendix D, Tables 10–13.

6 CONCLUSION

DUAL-MEM starts from the observation that an execution failure does not always invalidate the procedure being reused. Contextual intervention is more effective than committing to either Strategy revision or Card replacement, while cross-task improvement depends on retaining useful procedures together with evidence from their previous use. Cards also persist across multiple Strategy updates in the execution traces, matching their role as the more stable procedural state. Local execution failures should first be resolved at the level at which they occur, while changes to reusable memory should be reserved for evidence that persists beyond the current execution.

Reproducibility statement

The main text defines Card formation, the Fast state and actions, and segment utility. For LinUCB, each arm uses $\hat{\theta}_a = A_a^{-1} h_a$ and, after a segment closes, the standard updates $A_a \leftarrow A_a + x_k x_k^\top$ and $h_a \leftarrow h_a + y_k x_k$. At the first lifecycle event, no update is pending because the initial Card and Strategy precede the Controller's first decision. Qwen3.5-27B and Mistral-Small-3.1-24B-Instruct are served locally on one NVIDIA A800 GPU; GPT-5.6 Luna is accessed through its hosted interface. The appendices report benchmark statistics, Controller features, lifetime statistics, and parameter sweeps. Task orders, frozen Bank snapshots, execution scripts, and task-level results will accompany the submission as anonymous supplementary material.

AI use statement

Generative AI tools assisted with English translation, language polishing, and LaTeX preparation. The authors designed the method and experiments, collected the results, and reviewed the final manuscript.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. URL https://arxiv.org/abs/2204.01691.

Alibaba Cloud. Synchronous api for text embedding. Alibaba Cloud Model Studio documentation, 2026. URL https://www.alibabacloud.com/help/en/model-studio/text-embedding-synchronous-api. Documents the `text-embedding-v4` model.

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

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.866. URL https://aclanthology.org/2026.findings-acl.866/.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. URL https://arxiv.org/abs/2207.05608.

Eric Hanchen Jiang, Zhi Zhang, Yuchen Wu, Levina Li, Dong Liu, Xiao Liang, Rui Sun, Yubei Li, Edward Sun, Haozheng Luo, Zhaolu Kang, Aylin Caliskan, Kai-Wei Chang, and Ying Nian Wu. Memory as a controlled process: Learned adaptive memory management for llm agents. *arXiv preprint arXiv:2607.13591*, 2026. URL https://arxiv.org/abs/2607.13591.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017. doi: 10.18653/v1/P17-1147.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pp. 661–670, 2010. doi: 10.1145/1772690.1772758.

Gregoire Mialon, Clementine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. *arXiv preprint arXiv:2311.12983*, 2023. URL https://arxiv.org/abs/2311.12983.

Mistral AI. Mistral-small-3.1-24b-instruct-2503. Hugging Face model card, 2025. URL https://huggingface.co/mistralai/Mistral-Small-3.1-24B-Instruct-2503.

OpenAI. Gpt-5.6 luna. OpenAI API model documentation, 2026a. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna.

OpenAI. Gpt-5.6 sol. OpenAI API model documentation, 2026b. URL https://developers.openai.com/api/docs/models/gpt-5.6-sol.

Qwen Team. Qwen3.5-27b. Hugging Face model card, 2026. URL https://huggingface.co/Qwen/Qwen3.5-27B.

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://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *arXiv preprint arXiv:2305.16653*, 2023. URL https://arxiv.org/abs/2305.16653.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024. doi: 10.18653/v1/2024.acl-long.850.

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*, 2023a. URL https://arxiv.org/abs/2305.16291.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Cote, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.775.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. In *Advances in Neural Information Processing Systems*, volume 36, 2023b. URL https://arxiv.org/abs/2302.01560.

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. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

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, pp. 20004–20031, 2025. doi: 10.52202/085713-0593.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

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.

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

Bowen Ye, Yongchao Xu, Zhijian Li, Xiang Yin, Junkai Ma, and Wenzhao Li. Coevo-mem: Co-evolving retrieval policy and memory bank for llm agents. *arXiv preprint arXiv:2608.01739*, 2026. URL https://arxiv.org/abs/2608.01739.

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.

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.

A Benchmark and Evaluation Statistics

The evaluation protocol covers six benchmarks with distinct interaction patterns and evaluation granularity. Every benchmark maintains its own Memory Bank and Controller state so that procedural experience never transfers across benchmark boundaries. Source data used for memory construction remain disjoint from evaluation instances. Methods compared under the same seed receive an identical task order. Their execution budget and tool interface remain fixed, while scoring is delegated to the official evaluator of the corresponding benchmark. Aggregate scores use macro averaging across benchmarks, which prevents larger evaluation sets from dominating the reported mean. Detailed split information and evaluation counts are reported in Table 3.

Table 3: Evaluation statistics for the six benchmarks used in the main experiment.

Benchmark	Evaluation split	Instances	Task setting
ALFWorld	Unseen evaluation	134	Household interaction
AppWorld	`test_normal`	168	API and application interaction
ScienceWorld	Evaluation variations	270	Scientific experimentation
τ-bench	Airline and retail evaluation	164	Tool use under user interaction
TriviaQA	Wikipedia validation	7,993	Knowledge question answering
GAIA	Public validation	165	Multi step general assistance

Baseline Reproduction and Memory Construction. All memory-based methods are reproduced from their published specifications and receive the same benchmark-specific source pool before evaluation. Source tasks remain disjoint from evaluation instances, so the methods differ in how they represent and process prior experience rather than in the source information available to them. Each baseline retains its method-native memory representation, construction prompt, and online update mechanism, with published default hyperparameters used unless otherwise specified. We do not equalize the number of stored memory units because reflections, distilled experiences, skills, workflows, and procedural memories constitute different representation units. DUAL-MEM uses GPT-5.6 Sol (OpenAI, 2026b) only for Card extraction; competing methods use their respective

reproduction prompts and do not receive the DUAL-MEM Card representation or extractor. DUAL-MEM uses its Slow Lifecycle with the default configuration reported in Table 6. Thus, all memory methods share the same source pool, while each method retains the representation, construction and update rules, and published default hyperparameters of its original paper.

DUAL-MEM Initial Memory Bank. The shared-source protocol controls the pre-evaluation experience available to all memory-based methods while preserving each method's native representation. DUAL-MEM specifically begins each benchmark with a 60-Card Memory Bank constructed exclusively from successful source trajectories. GPT-5.6 Sol serves as the Memory Bank extractor throughout source construction. The extractor receives the task description together with the completed successful trajectory and produces one applicability description ω_i with one reusable procedure c_i. Applicability is expressed as a declarative condition describing when the procedure should transfer to a later task. The procedure is compressed into three to seven ordered steps and removes task-specific entities whenever the successful interaction supports a reusable abstraction. Extraction uses temperature zero. Candidate Cards enter the initial Bank only after satisfying the same semantic novelty rule used during online formation, with $\epsilon = 0.85$. Source trajectories are processed in a seed-fixed order until 60 Cards have been admitted. Evaluation instances never contribute to the construction of B_0.

Additional ReMe Comparison. ReMe is additionally reproduced on AppWorld, a benchmark shared with its original evaluation, under the same source-pool and execution protocol used for the other baselines. Across three independent runs, ReMe obtains $41.87 \pm 0.91\%$ success, compared with $41.27 \pm 0.91\%$ for MemCon and $44.05 \pm 0.60\%$ for DUAL-MEM.

Table 4: Additional comparison with ReMe on AppWorld under the shared-source evaluation protocol. Values are mean $\pm$ standard deviation over three independent evaluation runs.

Method	AppWorld Success (%) ↑
MemCon	41.27$_{(\pm 0.91)}$
ReMe	41.87$_{(\pm 0.91)}$
DUAL-MEM	**44.05**$_{(\pm 0.60)}$

DUAL-MEM Initial Usage State Persistent usage statistics start from an uninformative prior after the source Bank has been built. Each Card therefore enters evaluation with $n_i^+ = 0$, $n_i^- = 0$, and $f_i = 0$. The symmetric prior uses $\alpha_0 = \beta_0 = 1$, which initializes reliability at $u_i = 0.5$ while evidence support remains $v_i = 0$. Success of the source trajectory establishes that a Card may be formed, but it does not count as later Card usage. This separation ensures that target side reliability reflects actual reuse after the Card exists instead of counting its source trajectory twice.

Causal Evaluation Order Evaluation follows a causal task order. During task q_n, the persistent Bank and Usage State remain fixed. Only after termination can the resulting Task Record update persistent state for task q_{n+1}. Future evaluation tasks are unavailable to the current execution. Dynamic baselines retain their native update mechanisms under this same ordering, while static baselines preserve their initial memory state. The controlled Slow Lifecycle study uses a separate 120 task training stream before any probe evaluation occurs. The resulting memory state is frozen and then assessed on the same 134 task ALFWorld probe for every variant.

Execution and Token Accounting Interactive episodes are capped at 50 environment steps. Strategy generation and action generation use the execution backbone assigned to the corresponding experiment with separate prompts and temperature set to zero. The Strategy Generator conditions on the current task and recent execution evidence, while the active Card is supplied whenever memory guidance is available. The Actor then produces one environment action from the current Strategy at each step. For locally served models, the serving configuration is fixed across runs. Hosted-model inference may still exhibit small run-to-run variation despite temperature-zero generation. Online input token accounting includes every Actor call together with Strategy regeneration and state aware reacquisition prompts. Memory Bank extraction by GPT-5.6 Sol and post task Slow Lifecycle

processing are excluded because both occur outside online execution. TriviaQA is treated as a single response benchmark, so it uses initial memory retrieval without activating the event driven Controller.

B Controller Context, Memory State, and Operational Rules

Controller Context At each lifecycle event k, intervention selection is conditioned on a nine dimensional observable state. A constant bias occupies the first coordinate. The remaining state summarizes how execution is progressing and how strongly the active procedural memory is supported by current and historical evidence. The implemented context is

$$x_k = [1, b_k, \bar{d}_k, \bar{\ell}_k^{\sigma}, \bar{\ell}_k^{m}, \bar{r}_k, \bar{\rho}_k, u_i, \bar{v}_i]^{\top}. \tag{10}$$

Repeated behavior is represented by b_k. Its value becomes one when the normalized signature of the current action matches an action from the preceding three environment steps and no observable progress occurred between those executions. The distance d_k records how many environment steps have elapsed since the latest progress event. Strategy age ℓ_k^{σ} begins when the current Strategy is instantiated, while Card age ℓ_k^{m} begins when the current Card becomes active. The count r_k increases after each RENEW performed under that Card. Count features are mapped to comparable scales using the common episode limit $T_{\max} = 50$ according to

$$\begin{aligned} \bar{d}_k &= \min\left(\frac{d_k}{T_{\max}}, 1\right), \quad \bar{\ell}_k^{\sigma} = \min\left(\frac{\ell_k^{\sigma}}{T_{\max}}, 1\right), \\ \bar{\ell}_k^{m} &= \min\left(\frac{\ell_k^{m}}{T_{\max}}, 1\right), \quad \bar{r}_k = \frac{r_k}{1 + r_k}. \end{aligned} \tag{11}$$

Recent Execution Context The recent history H_t retains the six most recent action and observation transitions. Event context ξ_k is obtained through deterministic serialization. The task goal and current observation provide the immediate state, while the retained history and Watcher trigger supply the execution evidence needed at the intervention boundary. Its construction therefore introduces no additional language model call. Every semantic comparison in DUAL-MEM uses `text-embedding-v4` (Alibaba Cloud, 2026). Embedding vectors are normalized to unit length before cosine similarity is computed. The same representation space is preserved across both lifecycles, which keeps execution time suitability scores directly compatible with the similarity values used for retrieval and memory admission.

Local Card Applicability Local applicability measures whether the active Card still matches the state reached during execution. Compatibility is computed from the event context and the applicability description stored in the Card according to

$$\rho_k = \cos\left(\mathrm{emb}(\xi_k), \mathrm{emb}(\omega_i)\right), \qquad \bar{\rho}_k = \frac{1 + \rho_k}{2}. \tag{12}$$

The transformed value $\bar{\rho}_k$ lies in the unit interval and enters the LinUCB context. Retrieval continues to rank Cards with the original cosine score. This design allows the Controller to use local compatibility as evidence for intervention while leaving semantic retrieval independent of the historical reliability statistics.

Long Term Usage Evidence Persistent evidence is maintained separately for every Card. The counters n_i^{+} and n_i^{-} accumulate positive and negative outcomes observed before task q_n, while f_i counts completed tasks in which the Card became active at least once. Reliability and evidence support are computed as

$$u_i = \frac{1 + n_i^{+}}{2 + n_i^{+} + n_i^{-}}, \qquad v_i = \log(1 + f_i), \qquad \bar{v}_i = 1 - \exp(-v_i) = \frac{f_i}{1 + f_i}. \tag{13}$$

The posterior mean u_i summarizes empirical reliability under the symmetric Beta prior. Evidence support is compressed through $\bar{v}_i$, which prevents frequently activated Cards from obtaining disproportionate numerical scale in the linear Controller state. Neither quantity changes during task q_n. Their next values are written only after task termination and first become visible during task q_{n+1}.

Watcher Rule Controller invocation is event driven. The default stall threshold is $h = 4$, so repeated ineffective behavior becomes actionable only after four consecutive environment steps without observable progress. A second trigger is an internal refresh request raised when the latest observation leaves the current Strategy without a task-relevant continuation. The signal is based on the current execution context and does not consume an environment step. Once an intervention has been applied, the stall counter returns to zero and repeat detection restarts from the new segment. Consequently, every later Controller decision is supported by execution evidence collected after the preceding intervention.

Card Suitability and Retrieval A single suitability rule governs whether stored procedural memory may guide execution. For any retrieval context z, Card m_i receives the semantic score

$$s_i(z) = \cos\left(\mathrm{emb}(z), \mathrm{emb}(\omega_i)\right). \tag{14}$$

A Card is eligible when

$$\mathrm{Suitable}(m_i, z) = \mathbb{I}[s_i(z) \geq \theta_{\mathrm{use}}], \qquad \theta_{\mathrm{use}} = 0.60. \tag{15}$$

The threshold $\theta_{\mathrm{use}} = 0.60$ is fixed before evaluation and shared across benchmarks. Its role is restricted to execution time eligibility. The novelty threshold ϵ instead regulates persistent memory growth, so a Card may be semantically valid for current use without satisfying the stronger distinctiveness condition required for adding new memory.

At task initialization, every Card in B_n is available for retrieval. The task goal and first observation form the initial query through deterministic serialization. Among Cards that satisfy the suitability gate, the highest semantic score determines the active memory. Exact ties are resolved by stable Bank order. Failure to find an eligible Card causes execution to begin without procedural memory, in which case the Strategy Generator uses only the task state and recent evidence.

Reacquisition reconstructs the query from the current task and event evidence rather than reusing the initial query. The available pool $\mathcal{I}_k$ contains only Cards that have not already been abandoned during the present task. Selection of REACQUIRE removes the current Card before a replacement is sought. Eligible candidates therefore satisfy

$$\mathcal{E}_k = \{i \in \mathcal{I}_k \setminus \{i_k\} \mid s_i(z_k) \geq \theta_{\mathrm{use}}\}. \tag{16}$$

A nonempty candidate set yields the Card with the largest $s_i(z_k)$, after which a new Strategy is instantiated from its procedure. Once REACQUIRE abandons a Card, that Card cannot return during the same task. By contrast, RENEW preserves the retrieval pool because the underlying procedure remains active. Exhausting the eligible pool switches execution into memory free mode. The first environment step taken under this state defines the memory gap point that may later support Slow Lifecycle formation.

Observable Progress Progress detection is implemented by a benchmark adapter that reads only observations exposed through the environment or tool interface. A progress event requires newly established evidence that resolves some part of the remaining task. The Controller receives a standardized binary segment signal so that benchmarks with denser native feedback do not produce larger utility values simply because they expose more intermediate scores. Segment progress is defined as

$$\Delta p_k = \mathbb{I}[\text{at least one observable progress event occurs after event } k]. \tag{17}$$

The benchmark specific evidence used to instantiate this common definition is summarized in Table 5. The adapter never reads hidden simulator state that is unavailable to the acting agent.

Table 5: Operational progress signals used by the Watcher and the segment utility.

Benchmark	Observable progress event
ALFWorld	A textual observation confirms a new task relevant object state, receptacle state, or completed manipulation that advances the unresolved goal.
AppWorld	A structured tool response confirms a required application state, a successful side effect, or a newly verified intermediate condition.
ScienceWorld	The environment reports an increased task score or an observation confirms a newly satisfied experimental condition.
τ-bench	A tool response confirms a required transaction state or resolves a user constraint that remained unsatisfied before the call.
TriviaQA	The Fast Controller is inactive because the task is evaluated as a single response interaction.
GAIA	A tool call produces a new verified artifact, evidence item, or state transition that resolves an outstanding subgoal.

Segment Utility Every Controller action is evaluated on the execution segment that follows it. A decision made at event k remains pending until another lifecycle event occurs or the task terminates. Successful termination during this interval sets δ_k to one. Continued failure to make progress sets ζ_k to one when the interval closes at another stall event or at unsuccessful termination. Intervention cost is normalized by assigning

$$\kappa_k = \begin{cases} 0.25, & a_k = \textsc{Renew}, \\ 1.00, & a_k = \textsc{Reacquire}. \end{cases} \tag{18}$$

The cost difference reflects the additional retrieval and Strategy generation required by REACQUIRE, while RENEW regenerates guidance under the current Card. Segment utility is then computed as

$$\begin{aligned} y_k &= \Delta p_k + \eta\delta_k - \lambda\kappa_k - \mu\zeta_k, \\ \eta &= 1.00, \qquad \lambda = 0.15, \qquad \mu = 0.50. \end{aligned} \tag{19}$$

The coefficients remain fixed across benchmarks after being chosen on development tasks drawn from training data. Restored progress supplies the principal positive signal, while successful completion contributes an additional reward of equal magnitude. Intervention cost remains deliberately smaller than progress reward. Continued stagnation receives a stronger penalty so that expensive recovery actions are not reinforced when they fail to change the execution state.

LinUCB Initialization and Update LinUCB maintains one arm for RENEW and one arm for REACQUIRE. Because the observable context has dimension nine, both arms begin each benchmark with the same regularized state

$$A_a = I_9, \qquad h_a = \mathbf{0}, \qquad \widehat{\theta}_a = A_a^{-1} h_a. \tag{20}$$

The exploration coefficient is fixed at $\alpha = 0.50$. Once the pending segment closes, only the arm that produced that segment receives the observed utility through

$$A_{a_k} \leftarrow A_{a_k} + x_k x_k^\top, \qquad h_{a_k} \leftarrow h_{a_k} + y_k x_k. \tag{21}$$

Controller parameters persist across the causal task stream of one benchmark and are reinitialized when evaluation moves to another benchmark. Exact score ties select RENEW because it incurs the lower normalized intervention cost. No update occurs at the first lifecycle event because no previous Controller decision exists. If a task terminates while an intervention remains pending, its final segment is closed and written to LinUCB before the Task Record enters the Slow Lifecycle.

Card Use Evidence Intervention utility and Card reliability are updated from the same execution trace but answer different questions. LinUCB evaluates whether the selected recovery action improved the following segment. Usage State instead evaluates whether the procedural Card remained useful while it was active. A closed segment under Card i contributes positive evidence when observable progress occurs or the task finishes successfully. Negative evidence is assigned only when no progress occurs and the Card is subsequently abandoned through REACQUIRE or remains active until unsuccessful termination. Failure followed by RENEW remains neutral for Card reliability because the framework first attributes that local failure to the Strategy realization. With g_s^+ and g_s^- denoting the segment labels, this rule becomes

$$g_s^+ = \mathbb{I}[\Delta p_s = 1 \ \vee \ \delta_s = 1]\,, \qquad g_s^- = \mathbb{I}[\Delta p_s = 0 \ \wedge \ D_s = 1]\,, \tag{22}$$

Here D_s equals one when REACQUIRE abandons the Card or when unsuccessful task termination occurs while the Card is still active. A segment with $g_s^+ = g_s^- = 0$ leaves the Beta counts unchanged.

Only completed tasks may modify persistent usage evidence. Once task q_n terminates, its segment labels update the Card specific counters through

$$n_i^+ \leftarrow n_i^+ + \sum_{s:i_s=i} g_s^+, \qquad n_i^- \leftarrow n_i^- + \sum_{s:i_s=i} g_s^-, \tag{23}$$

$$f_i \leftarrow f_i + \mathbb{I}[\text{Card } i \text{ was activated during task } q_n]\,. \tag{24}$$

Updated counts determine the next values of u_i and v_i, which first become visible during task q_{n+1}. Semantic retrieval still depends on ω_i through cosine similarity in the shared `text-embedding-v4` space. Usage history therefore changes the evidence available to the Controller without rewriting the retrieval representation.

Online Card Formation Persistent memory grows only when successful execution reveals procedural coverage that the existing Bank did not provide. A memory gap begins after initial retrieval rejects every Card or after REACQUIRE exhausts the available pool. The first environment step executed without an active Card is denoted by t_{gap}. When the task later succeeds, the suffix beginning at this point is treated as uncovered successful experience. GPT-5.6 Sol receives the original task goal together with the state at the memory gap and the successful suffix. Using the same extraction prompt and temperature zero as source Bank construction, it produces a candidate Card $\widetilde{m}$. The generated applicability description is denoted by $\widetilde{\omega}$, and its reusable procedure is denoted by $\widetilde{c}$.

Admission depends on semantic distinctiveness of the candidate applicability description relative to the persistent Bank. The shared `text-embedding-v4` representation is used for this comparison, and the candidate is accepted only when

$$\max_{m_i \in B_n} \cos\left(\mathrm{emb}(\widetilde{\omega}), \mathrm{emb}(\omega_i)\right) < \epsilon, \qquad \epsilon = 0.85. \tag{25}$$

A completed task can contribute at most one new Card. Existing Card text remains immutable during target evaluation, and deletion is disabled. Long term adaptation of an existing Card is represented through Usage State rather than textual rewriting. Newly admitted memory starts with the same symmetric prior used by the initial Bank. The successful suffix responsible for its creation does not count as later usage because the Card did not yet exist while that experience was collected.

Default Operational Configuration The complete operational specification is consolidated in Table 6. Sensitivity experiments alter only the designated Watcher threshold, memory admission threshold, and LinUCB exploration coefficient. Every other quantity remains fixed across the controlled configurations described in Appendix D.

Table 6: Default operational configuration used by DUAL-MEM.

Quantity	Default	Operational role
Initial Bank size	60 Cards	Shared starting memory capacity within a benchmark
Memory Bank extractor	GPT-5.6 Sol	Generates source Cards and online candidate Cards from successful trajectories
Embedding model	`text-embedding-v4`	Shared semantic representation used for memory retrieval and lifecycle similarity decisions
Maximum episode length $T_{\max}$	50 steps	Common execution budget for interactive tasks
Recent history window	6 transitions	Context used by Strategy generation and event serialization
Repeat detection window	3 steps	Window used to construct b_k
Watcher threshold h	4 steps	Consecutive stagnation required before a stall event
Suitability threshold θ_{use}	0.60	Minimum cosine similarity for active memory use
Admission threshold ϵ	0.85	Maximum allowed similarity for admitting a new Card
LinUCB exploration α	0.50	Weight assigned to uncertainty in intervention selection
LinUCB ridge initialization	1.00	Diagonal value in each initial A_a matrix
Beta prior α_0	1.00	Positive reliability pseudocount
Beta prior β_0	1.00	Negative reliability pseudocount
Completion coefficient η	1.00	Weight on terminal task success
Intervention coefficient λ	0.15	Weight on normalized intervention cost
Stagnation coefficient μ	0.50	Penalty for unresolved post intervention stagnation
RENEW cost	0.25	Relative cost of Strategy regeneration
REACQUIRE cost	1.00	Relative cost of Card replacement and Strategy regeneration
Card extraction temperature	0.00	Deterministic persistent memory formation
Execution temperature	0.00	Deterministic Strategy generation and Actor decoding

C CARD AND STRATEGY LIFETIME STATISTICS

Span Definition The lifetime analysis underlying Figure 5 uses 402 ALFWorld evaluation episodes from the default Qwen3.5-27B configuration. Strategy lifetime starts at instantiation and closes when RENEW replaces the Strategy, when REACQUIRE replaces the Card, or when the task terminates. Card lifetime starts at activation and closes only after REACQUIRE or task termination. Termination is treated as right censoring because the active state might have remained valid beyond the observed episode horizon.

Restricted mean survival time through 40 environment steps provides a common summary under censoring. Let $\widehat{S}(t)$ denote the Kaplan Meier estimate for the corresponding active state. The reported quantity is

$$\text{RMST@40} = \int_0^{40} \widehat{S}(t)\, dt. \tag{26}$$

Mean values and sample standard deviations across the three task order seeds appear in Table 7.

Table 7: Card and Strategy timescale statistics across three task order seeds.

Metric	Seed 1	Seed 2	Seed 3	Aggregate or mean
Episodes	134	134	134	402
Active Card spans	245	251	248	744
Active Strategy spans	443	458	450	1,351
RENEW events	198	207	202	607
REACQUIRE events	111	117	114	342
Strategy RMST@40	8.70	9.00	9.10	8.93(± 0.21)
Card RMST@40	19.60	20.30	20.90	20.27(± 0.65)
Card to Strategy RMST ratio	2.25×	2.26×	2.30×	**2.27**$(\pm 0.02)\times$
Strategies per active Card	1.81	1.83	1.82	1.82

Count Consistency Lifecycle transitions determine the observed span counts exactly. Every episode introduces one initial Card span together with one initial Strategy span. A REACQUIRE event starts a new Card span and simultaneously instantiates a new Strategy. A RENEW event creates a new Strategy while preserving the current Card. These identities yield

$$N_{\text{Card}} = N_{\text{episodes}} + N_{\text{REACQUIRE}} = 402 + 342 = 744, \tag{27}$$

$$N_{\text{Strategy}} = N_{\text{episodes}} + N_{\text{REACQUIRE}} + N_{\text{RENEW}} = 402 + 342 + 607 = 1,351. \tag{28}$$

Table 8: Distribution of successive Strategies within one active Card span.

Strategies per Card	Aggregate count	Fraction
1	349	46.90%
2	251	33.70%
3	90	12.10%
4	40	5.40%
5	14	1.90%

Across 744 observed Card spans, the mean number of Strategies per active Card is 1.82. Multiple Strategy realizations occur in 395 spans, corresponding to 53.10% of the total. These cases capture execution in which task specific guidance changes while the same procedural memory continues to support the task.

Table 9: Ending events for active Strategy spans and active Card spans.

Ending event	Strategy count	Strategy fraction	Card count	Card fraction
RENEW	607	44.90%	N/A	N/A
REACQUIRE	342	25.30%	342	46.00%
Task termination	402	29.80%	402	54.00%
Total	1,351	100.00%	744	100.00%

Interpretation Card RMST@40 reaches 20.27 ± 0.65 steps, while Strategy RMST@40 reaches 8.93 ± 0.21 steps. Their mean ratio is $2.27 \pm 0.02\times$ across seeds. This difference describes the temporal behavior induced by the DUAL-MEM control structure. A RENEW decision closes the current Strategy span by construction, whereas the Card remains active until replacement or task termination. The measurements therefore quantify how often local guidance changes within a more persistent procedural state. They should be interpreted as lifecycle dynamics rather than an independent causal validation of the hierarchy.

D HYPERPARAMETER SENSITIVITY

Sensitivity Protocol Sensitivity experiments modify one control variable at a time while preserving every other value in Table 6. Each configuration is evaluated with three task order seeds on the same 134 task ALFWorld set. Changing the admission threshold requires the full 120 task continual stream to be rerun from the shared 60 Card initial Bank before the probe state is frozen. Watcher and LinUCB sweeps follow the same continual protocol used by the default configuration. Consequently, every default point refers to the same system state and the same execution logs analyzed in Appendix C.

Robustness is summarized with a stable region defined as the largest contiguous interval that contains the default setting and remains within one percentage point of its mean success. If $S(\phi)$ denotes mean success under parameter value ϕ and ϕ_0 denotes the default, admissible values satisfy

$$\mathcal{R}_\phi = \{\phi \,|\, S(\phi) \geq S(\phi_0) - 1.00\}\,. \tag{29}$$

The resulting regions and tested ranges are collected in Table 10.

Table 10: One at a time sensitivity ranges and stable operating regions around the 73.88% default.

Component	Parameter	Tested range	Default	Stable region	Success range
Watcher	h	2.00 to 8.00	4.00	$[3.00, 5.00]$	73.13 to 73.88
Card admission	ϵ	0.75 to 0.95	0.85	$[0.80, 0.90]$	72.89 to 73.88
LinUCB	α	0.00 to 1.60	0.50	$[0.20, 0.80]$	72.89 to 73.88

Watcher Threshold Watcher sensitivity reveals how much execution evidence should accumulate before recovery begins. Small h values react quickly and therefore create more lifecycle events, whereas large values permit the current Strategy to persist through longer periods without progress. The detailed sweep in Table 11 places the default $h = 4$ inside the stable region from 3 to 5. Across this interval, mean success varies by at most 0.75 percentage points while event frequency falls from 2.91 to 1.98 events per task.

Table 11: Sensitivity to the Watcher stall threshold across 402 evaluation episodes.

h	Success %	Steps	Input tokens per task k	Total events	Events per task	RENEW share	REACQUIRE share
2.00	71.39($\pm$1.88)	21.70($\pm$0.26)	45.43($\pm$0.31)	1,503	3.74	73.99%	26.01%
3.00	73.13($\pm$1.49)	21.13($\pm$0.25)	42.80($\pm$0.30)	1,170	2.91	68.97%	31.03%
4.00	**73.88**($\pm$1.49)	**20.90**($\pm$0.40)	**41.90**($\pm$0.26)	**949**	**2.36**	**63.96%**	**36.04%**
5.00	73.38($\pm$1.55)	21.40($\pm$0.40)	42.33($\pm$0.21)	796	1.98	60.05%	39.95%
6.00	71.89($\pm$1.55)	22.50($\pm$0.36)	43.43($\pm$0.35)	667	1.66	55.02%	44.98%
8.00	69.65($\pm$1.55)	24.07($\pm$0.31)	45.93($\pm$0.35)	515	1.28	47.96%	52.04%

Behavior outside the stable interval explains the loss in success. With $h = 2$, intervention occurs before enough evidence has accumulated and online token use rises. With $h = 8$, stalled execution persists longer before recovery and mean episode length reaches 24.07 steps. Larger thresholds also shift decisions toward REACQUIRE, which is consistent with later events carrying stronger evidence that the active procedure itself may no longer be suitable.

Memory Admission Threshold Memory admission sensitivity characterizes the balance between procedural coverage and redundancy. Lower values of ϵ demand greater semantic separation before a candidate can enter the Bank. Higher values accept candidates that are closer to stored applicability descriptions. Every setting in Table 12 begins from the same 60 Card Bank and repeats the full 120 task continual stream. The default $\epsilon = 0.85$ produces a mean final Bank size of 76.67 Cards together with the highest mean probe success. Values from 0.80 through 0.90 remain inside the one percentage point stability criterion.

Table 12: Sensitivity to the Slow Lifecycle admission threshold and the resulting final Bank size.

ϵ	Success %	Steps	Input tokens per task k	Final Banks	Mean final Bank	New Cards
0.75	71.14($\pm$0.86)	22.40($\pm$0.20)	43.00($\pm$0.20)	67 / 68 / 69	68.00($\pm$1.00)	8.00
0.80	72.89($\pm$1.14)	21.53($\pm$0.15)	42.27($\pm$0.25)	71 / 72 / 73	72.00($\pm$1.00)	12.00
0.85	**73.88**($\pm$1.49)	**20.90**($\pm$0.40)	**41.90**($\pm$0.26)	**75 / 76 / 79**	**76.67**($\pm$2.08)	**16.67**
0.90	73.38($\pm$1.14)	21.20($\pm$0.36)	41.83($\pm$0.21)	82 / 84 / 86	84.00($\pm$2.00)	24.00
0.95	71.64($\pm$1.49)	22.13($\pm$0.25)	43.10($\pm$0.30)	93 / 96 / 99	96.00($\pm$3.00)	36.00

Increasing ϵ expands the mean final Bank from 68.00 to 96.00 Cards, yet probe success does not increase monotonically. Strict admission limits coverage because potentially useful procedures are rejected. Permissive admission increases semantic overlap among stored applicability descriptions and creates stronger competition during retrieval. The default value therefore occupies a middle regime where new procedural coverage can accumulate without allowing Bank growth to become dominated by near duplicate memory.

LinUCB Exploration Coefficient LinUCB sensitivity measures how aggressively the Controller values uncertainty when choosing an intervention. Setting $\alpha = 0$ removes the exploration bonus

and leaves decisions to the current utility estimates. Larger values assign more weight to uncertain actions and therefore expose less explored intervention contexts more often. The sweep summarized in Table 13 identifies a stable region from 0.20 through 0.80 around the default $\alpha = 0.50$.

Table 13: Sensitivity to the LinUCB exploration coefficient.

α	Success %	Steps	Input tokens per task k	Total events	RENEW share	REACQUIRE share
0.00	71.39($\pm$1.14)	22.13($\pm$0.25)	42.50($\pm$0.30)	872	69.04%	30.96%
0.20	72.89($\pm$1.14)	21.50($\pm$0.30)	41.93($\pm$0.25)	915	66.01%	33.99%
0.50	**73.88**($\pm$1.49)	**20.90**($\pm$0.40)	**41.90**($\pm$0.26)	**949**	**63.96%**	**36.04%**
0.80	73.38($\pm$1.55)	21.20($\pm$0.30)	42.17($\pm$0.25)	984	60.98%	39.02%
1.20	71.64($\pm$1.49)	22.03($\pm$0.25)	42.97($\pm$0.25)	1,028	57.00%	43.00%
1.60	70.15($\pm$1.49)	23.10($\pm$0.30)	44.10($\pm$0.30)	1,076	52.04%	47.96%

The share of REACQUIRE decisions rises from 30.96% at $\alpha = 0$ to 47.96% at $\alpha = 1.60$. Moderate exploration gives the Controller enough opportunity to learn when Card replacement restores progress. At larger values, uncertainty receives enough weight to promote the more expensive Card level intervention even when its estimated utility is weaker. The accompanying increase in token cost and decrease in success at 1.20 and 1.60 are consistent with this over exploration regime.

Sensitivity Summary Taken together, the default system is not tied to an isolated parameter setting. Each default lies inside a contiguous region whose mean success remains within one percentage point of the default despite changes in intervention timing, Bank growth, and uncertainty weighting. This stability supports one shared operational configuration throughout the controlled analyses.