
Checking Failures Before Updating Agent Memory: A Controlled Study

Neeraj Venna

Abstract

I test whether checking why an action failed helps an agent decide what to change in memory. The experiment compares symbolic Python controllers in a gate simulator and a local database. Alice keeps several possible explanations, chooses checks by estimated benefit and cost, and updates individual memory fields. Across 1,920 main-evaluation episodes per controller, Alice completed 46.6%, compared with 50.8% for action verification and 44.6% for immediate revision. Its difference from action verification was -4.2 percentage points (paired 95% interval $[-5.8, -2.7]$). Alice made fewer incorrect memory edits, but also marked more correct knowledge uncertain. Choosing targeted checks showed no clear overall completion benefit. All controllers failed when observations stayed stale because their shared planner still required confirmation from the unreliable channels. In this study, fewer memory errors did not produce the best task performance. The results point to a limitation of the confirmation rule, not a general benefit from diagnosis before revision.

1 Introduction

A failed gate-opening attempt can have several causes: the credential changed, the tool failed, or the gate opened but the observation is stale. These explanations call for different memory updates.

A check can help distinguish these causes, but it can also return the wrong answer and use up the task budget. For example, changing a remembered credential may fix an access-rule change while making a tool failure harder to handle. An agent might also complete the task by trying another tool without knowing why the first failed. I test whether checking the cause improves the eventual decision.

I compare memory policies in two small environments that keep their state between actions. One controls gates. The other updates database accounts and creates transaction receipts. In the database task, an update can succeed without creating its receipt, so a correct account balance does not necessarily mean the work is complete.

Alice reduced incorrect memory edits, but action verification completed more episodes and retained more correct knowledge as supported. Persistent stale observations defeated every controller: all shared a confirmation rule that continued to require the unreliable channel.

The controllers are hand-designed symbolic programs. Only a scalar observation-error estimate is learned. The study evaluates these memory policies and their failure modes; it does not evaluate language-model agents or learned neural world models.

2 Environments

2.1 Two persistent environments

An episode consists of 24 orders involving four targets, each visited repeatedly. An order asks for a gate position or an account version. The controller can use either of two tools, two credentials, and two observation channels. It can reuse information from earlier orders, but cannot reset or rewind the environment.

In the **gate environment**, a correct credential and working tool set the requested position. A failed tool does nothing. Both observation channels normally report whether the requested position has been reached. Gates retain their positions across actions; this is remote gate control, rather than a navigation task.

The **database environment** runs local SQLite transactions in a disposable database. A completed order needs an account update and a linked receipt. One tool can fail after writing only the account; the other writes both records together and can add a missing receipt without duplicating the transaction. The primary observation channel checks only the account. The secondary checks the account, receipt, and transaction key. Every controller knows this difference, and the evaluator always requires the complete transaction.

The controller chooses when to accept an order and move on. It receives no correctness feedback when it does so. An incorrect acceptance remains incorrect for scoring even if later actions repair the underlying state. **Episode success requires all 24 orders to be correctly accepted.** Unfinished work and acceptance of incomplete work both count as failure.

2.2 Actions and failure conditions

Table 1 lists the operations. Inspecting a credential or testing a tool costs 2 units at the low read price and 6 at the high price. A stale channel may repeat the same wrong answer, so repeated reads need not provide independent evidence. Tool tests also change the environment by writing calibration records. Diagnostic policies can inspect credentials, test tools, or read the current target through the secondary channel; none can ask the evaluator for the cause.

Operation	Purpose	Cost (units)
Apply	Try the requested update using a tool and credential	2
Read	Check through one observation channel	c
Inspect	Read a credential rule through a fallible channel	$2c$
Test	Exercise a tool on a calibration record	$2c$
Advance	Accept the current order and move to the next	1

Table 1: Every operation consumes the episode budget, which is 160 or 240 units. The read price c is either 1 or 3. Changing c changes mandatory confirmation reads as well as optional checks.

The experiment tests three sources of failure. A **world change** changes the required credentials on two targets. A **capability change** makes one tool fail; the other remains available. An **observation change** freezes a channel’s cached answers. Ordinary random bit-flip noise can affect checks in every condition. Stale memory is a possible consequence of these failures or of a mistaken update, rather than a fourth physical cause. Here, a self-model records available tools and their capabilities.

The main evaluation includes no change, each type of change alone, repeated changes, and both observation channels freezing at once. In repeated-change episodes, the previous failure is repaired before the next begins. Separate tests combine two failure types or introduce a cause absent from the controller’s model. In that last test, execution reverses which credential works, but inspection continues to report the original rule.

The simulator caches answers even for questions the agent has not asked. Changes occur at scheduled points in cumulative interaction cost, independent of how often a policy samples random numbers. Agents see tasks, their actions and responses, elapsed cost, and the order index. Hidden state and change labels are available only to the evaluator. Appendix A specifies this information boundary.

3 Controllers

3.1 Shared planner and comparison policies

Every controller receives the same initial knowledge, raw observation history, action interface, and interaction limits. A shared planner ranks the eight tool/credential/channel combinations using memory and avoids repeating failures on the current order. It accepts an order after two consecutive positive apply/read checks, switching channels after the first positive check. Diagnostic checks cannot interrupt this confirmation step. Using one planner isolates differences between memory policies, but also gives them a common weakness when observations stay stale.

The seven main controllers differ in how they use evidence:

- **Full history** recomputes success frequencies for matching targets and action paths from the complete observation history. It is a specific symbolic use of history, not unrestricted reasoning.
- **Immediate revision** changes memory in response to failure or new evidence. **Recency revision** waits for a second failure before making a failure-driven edit.
- **Periodic probing** adds a short fixed sequence of checks.
- **Field verification** checks task-relevant memory fields according to age, uncertainty, and a criticality rule.
- **Action verification** checks when its model predicts that the evidence could reduce the risk of the next physical action failing, then commits to the highest-weight joint explanation.
- **Alice** also values evidence that could change a memory decision, and can defer a revision rather than commit to one explanation.

Field and action verification are limited symbolic adaptations of published methods, not reproductions of those systems. Each main policy had the same development search over two configurations. The field-verification adaptation has a costly inspection schedule, analyzed below.

3.2 Alice memory policy

Memory starts with eight correct entries: four credential rules, two tool-health values, and two channel-reliability values. Entries store a value, timestamp, supporting observations, confidence, and a status of supported or uncertain. Raw observations are stored separately. When a value changes, its old and new values are retained. Confirming an unchanged value is not counted as an edit.

Alice considers eight combinations of world, tool, and observation faults, plus an “unknown” explanation. It assigns a weight to each combination. A failed response can leave several explanations plausible, such as a broken tool or a stale observation. These weights come from a hand-designed approximate model; calibration has not been established. Alice keeps the combinations while working on an order, then uses persistent memory to initialize the next one. It does not retain all correlations between orders.

Only the ordinary observation-error probability is estimated from data. Training produced 58 mismatches in 1,600 observations from healthy environments. Adding one error and one correct observation gives the shared estimate $59/1602 = 0.03683$. The expected failure patterns, change assumptions, thresholds, and unknown explanation are hand-designed. This is not a learned general world model.

Alice records a prediction before each response arrives. A zero response after an apply action makes it consider a diagnostic check; hidden change labels never trigger a check. It predicts whether each candidate could reduce memory errors or the chance that the next action fails, then subtracts a cost penalty. A check is taken only if this estimated benefit is positive and both the budget and per-order limit permit it. The rule looks one step ahead rather than optimizing the whole episode (Appendix B).

Alice changes an affected memory field only when its marginal support—the total weight favoring that value—reaches 0.80. Otherwise it marks the field uncertain. Enough weight on the unknown

explanation makes all three affected fields uncertain; unrelated entries stay unchanged. The planner can still use an uncertain value to rank actions, but the retention metric does not count that value as reliably retained knowledge.

4 Evaluation

4.1 Evaluation population and uncertainty

I used development data to select settings and estimate variability. The final evaluation holds out object identities, order sequences, change times, noise values, and some corruption parameters, while keeping the same small task structure. Before the corrected evaluation, I fixed the code, parameters, settings, and analysis rules in a versioned local snapshot. This was a local freeze, not an independently witnessed preregistration.

The full evaluation contains **38,400 controller episodes**: 2,560 scenarios evaluated by 15 controllers, including controlled variants and references. The main population has **1,920 episodes per controller**. It crosses two environments, six change conditions, two budgets, and two read prices, giving 48 equally weighted conditions. Compound changes and the missing-explanation failure each have another 320 episodes per controller and are reported separately.

The evaluation uses **40 independent base scenario units** in eight batches of five. Each base unit generates related cases across conditions, and every controller receives those same cases. These 40 base units determine the resampling: the 38,400 episodes cannot be treated as independent trials. Each of 2,000 bootstrap draws resamples whole base units, keeping all associated methods and conditions together. Reported 95% intervals apply to individual comparisons; they do not provide 95% simultaneous coverage across all comparisons. Differences in percentage points are absolute differences, not relative changes.

An earlier run stopped after 8,714 saved episodes because its first change always occurred at time 11 instead of the specified 11 or 12. The amendment written at the time records that this was found before performance summaries were inspected. Those records are preserved but excluded from the results. The corrected run changed the timing and seed namespace, without retuning policies or changing outcomes, budgets, or sample size. All reported results use the fresh corrected holdout; Appendix C gives the protocol history and sample-size calculation.

4.2 Task, memory, and recovery measurements

The primary outcome is the fraction of episodes in which all 24 orders are accepted correctly under the assigned budget. Incomplete episodes and software failures remain in the denominator. The completed evaluation had zero execution errors; ordinary task failures are retained.

An **incorrect memory edit** is an accepted value change that disagrees with evaluator truth immediately afterward. The code calls this a “harmful edit,” but it does not show that the edit caused worse behavior. **Collateral damage** is an incorrect edit to a value that was correct before the action and whose underlying truth did not change. Mean counts include episodes with no edits. **Edit coverage** is the fraction of episodes with at least one accepted value-changing edit.

Supported retention measures whether correct knowledge stays both correct and trusted. Before each action, the evaluator selects memory fields whose stored value is correct and whose true value does not change during that action. Each field/action pair counts as one opportunity. It is retained only if the stored value remains correct and supported after the response. The analysis pools opportunities within each condition, then averages the condition rates equally. Correct values marked uncertain do not count. This measures neither overall memory accuracy nor the fraction of distinct memories retained.

The **conditional incorrect-edit rate** is the fraction of accepted edits that are wrong. It is computed within each condition, then averaged equally across conditions where at least one edit occurred. Dividing the aggregate mean counts gives a different weighting. A policy that never edits has an undefined rate, so the results also report edit counts and coverage.

First-progress recovery asks whether the agent correctly accepts an order within 16 cost units after a hidden change. Observation stops at the next change or the episode’s end if either occurs

earlier. Windows without progress are retained as censored cases. An acceptance can concern work physically completed before the change, so this metric does not necessarily show sustained recovery. Appendix C.1 defines attribution and prediction scores separately.

5 Results

5.1 Task completion

Alice completed 46.6% of episodes, compared with 50.8% for action verification (Table 2). The paired difference was -4.2 percentage points, with a 95% interval of $[-5.8, -2.7]$. Alice exceeded immediate revision’s 44.6% by 2.0 points, interval $[0.6, 3.4]$. It therefore improved on immediate revision, but not on the strongest main baseline.

Controller	Success (%)	Incorrect edits	Retention (%)
Immediate revision	44.6	14.63	97.3
Action verification	50.8	3.38	99.5
Alice	46.6	0.31	90.5

Table 2: The main trade-off, over 1,920 episodes per controller and 48 equally weighted conditions. Success requires all 24 orders. Incorrect edits are means per episode. Retention measures correct, unchanged field/action opportunities that remain correct and supported; it is not memory accuracy. Paired success intervals are given in the text.

The benefit over immediate revision depended on the environment. In database tool-failure cases, Alice completed 74.4%, immediate revision 11.9%, and action verification 78.8%. A successful account read can conceal a missing receipt, so changing a credential belief does not fix the transaction. In gate world-change, tool-failure, and repeated-change cases, Alice instead trailed immediate revision. These setting-specific observations are exploratory; Appendix D shows every regime. They do not establish an advantage across failure causes.

Checks also used resources that could otherwise finish work. Figure 1 separates the four budget-and-price conditions. At budget 160 and read price 3, Alice completed 12.7%, compared with 23.5% for action verification and 18.8% for immediate revision. At the same budget and price 1, the rates were 58.5%, 59.4%, and 54.2%. Price affects required reads as well as optional diagnostics, so this comparison does not isolate the price of diagnosis. Overall Alice used 164.4 units per episode; action verification used 161.1.

The full main comparison and five planned contrasts appear in Appendix D. Immediate and recency revision had identical success despite different edit counts. The field-verification adaptation performed poorly partly because its inspection schedule could exceed the budget even without errors. Its large deficit is therefore not evidence against environment probing in general (Appendix D.1).

5.2 Memory errors and retention

Alice made 0.31 incorrect edits per episode, compared with 3.38 for action verification and 14.63 for immediate revision. Alice’s 95% interval is $[0.25, 0.37]$. Its collateral-damage count was also lower. These are changes to assertions, not proven effects on later task performance.

Alice also stopped treating more correct values as reliable. Supported retention was 90.5%, compared with 99.5% for action verification and 97.3% for immediate revision. Alice changed at least one value in 86.2% of episodes, versus 97.9% and 98.8%. Deferring an edit can avoid a wrong assertion while leaving a correct value uncertain. Lower edit error did not translate into better retention or completion.

The condition-averaged incorrect-edit rates were 31.1% for Alice, 45.4% for action verification, and 49.9% for immediate revision. All 48 conditions had nonzero edit denominators for these methods. For Alice, pooling all edits instead gives 591 incorrect edits out of 2,656 accepted edits, or 22.3%. The reported 31.1% gives equal weight to conditions; the pooled 22.3% weights conditions by their edit counts. Dividing the table’s mean counts estimates the latter quantity, even without rounding. Appendix C.1 gives the exact formula.

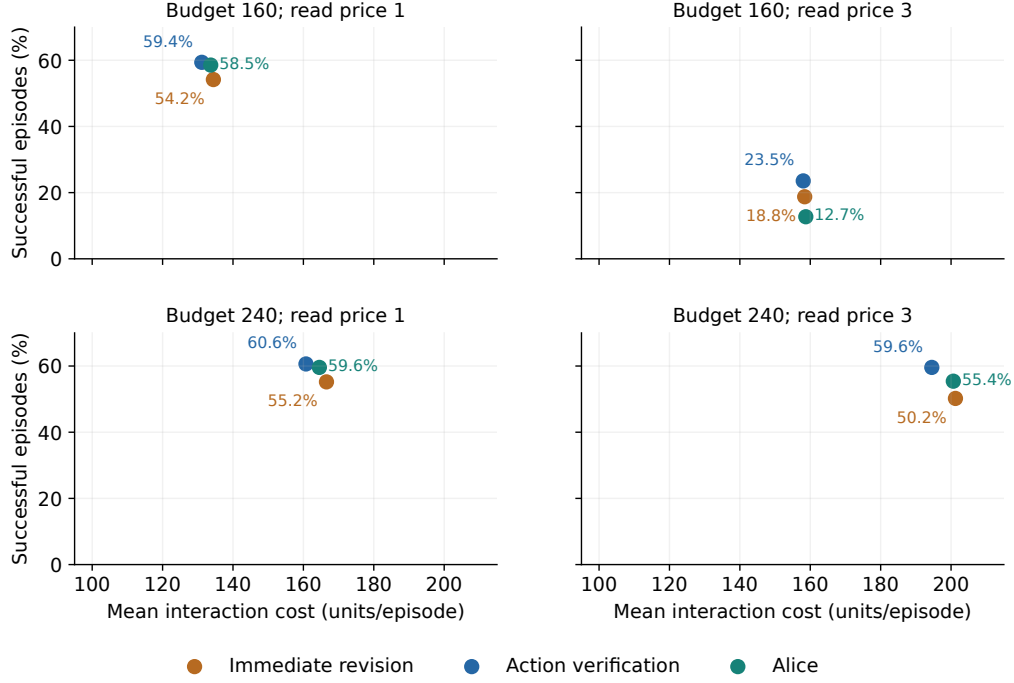


Figure 1: Cost and completion at each budget and read price. Points average 12 primary environment/change conditions, with 40 episodes per condition, for the three controllers in Table 2. Labels show success percentages. No curve is fitted, and uncertainty bars are omitted; paired intervals are reported in the text. Read price changes mandatory confirmation costs as well as optional diagnostic costs.

5.3 Controlled variants

A planned secondary comparison varied two choices: whether to select a targeted or random check, and whether to revise fields selectively or immediately commit to one joint explanation. The random-check variant kept the same rule for deciding whether to check at all. The variants thus shared opportunities and limits, but could make different numbers of checks and incur different costs.

The average targeting contrast was +0.10 percentage points, with a 95% interval of $[-0.55, 0.73]$. The interval provides no clear evidence of an aggregate benefit and does not establish equivalence. Selective revision reduced incorrect edits by 3.07 per episode but also reduced success by 2.55 points, interval $[-3.80, -1.33]$. Table 5 gives the planned contrasts.

Alice differed from the information-gain variant by +0.05 points, interval $[-1.04, 1.15]$. Collapsing the explanation weights to one alternative after every response reduced success to 33.4%. Keeping alternatives helped relative to this particular commitment rule. The comparison does not show that Alice’s check-selection score was better than the alternative scores. Appendix D reports all variants.

6 Failure analysis

6.1 Stale observations

Every controller completed zero episodes in both persistent observation-corruption conditions. Execution itself did not fail. The planner still required confirmation from the other channel even after memory marked that channel unreliable. Reliability beliefs could affect which action route it tried, but not whether it accepted an order. A stale negative answer could therefore prevent any further progress.

I inspected all saved Alice traces for these conditions after the main analysis. In all 320 isolated-observation episodes, Alice correctly marked the primary channel unreliable but still failed to finish. This establishes that it detected that unreliable channel, not that it identified every fault. No alternative confirmation rule was tested, so the result identifies a limitation of this planner rather than an impossible task.

6.2 Recovery measurement

Alice’s main first-progress recovery score was 94.8%, compared with 95.6% for action verification and 95.0% for immediate revision. These high values did not imply sustained adaptation. In the exploratory sensor-trace audit, 294 of 302 isolated-channel recovery events accepted work physically completed before the change. For both-channel corruption, the count was 294 of 296. In total, 588 of 598 recorded events accepted pre-change work.

Figure 2 compares first progress with complete-episode success in the observation-failure cases. I retain the original recovery outcome and label the later trace inspection as exploratory. The two measurements answer different questions.

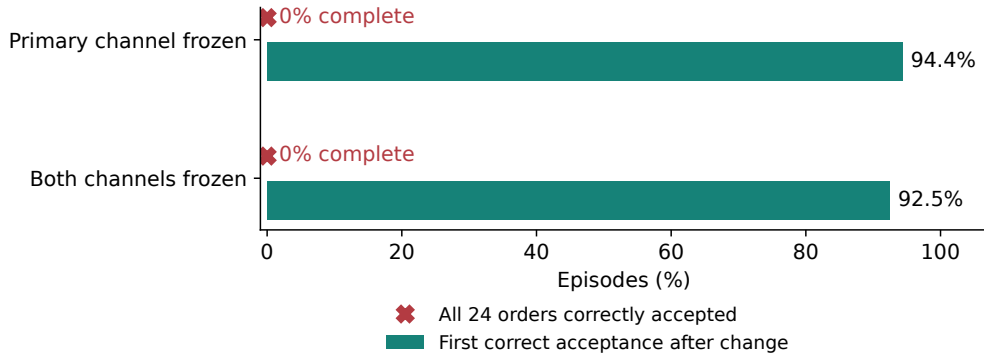


Figure 2: First progress and full completion under persistent observation faults. Each row averages eight environment/budget/price conditions and 320 episodes. Bars measure a correct acceptance within the fixed post-change window; crosses measure all-orders success. Most progress events accepted work already finished before the change. No uncertainty intervals are shown. High first-progress scores do not establish adaptation.

6.3 Unmodeled failure

When execution reversed the credential mapping without changing the inspected rule, Alice completed 35.3%, compared with 72.5% for action verification and 77.8% for immediate revision. These separate robustness results use 320 episodes per controller. The unknown explanation did not discover the missing mechanism. Alice’s supported retention fell to 64.3%, and its conditional incorrect-edit rate rose to 97.0%, despite only 1.65 incorrect edits per episode.

This condition also shows why an incorrect edit is not automatically a harmful action. A controller can hold a credential belief that is false according to the inspected rule yet choose an effective credential because execution reverses it. Simple adaptation can compensate for a wrong model without identifying the cause. In the separately tested compound conditions, Alice completed 29.7%, between action verification’s 35.0% and immediate revision’s 24.4%. Neither test supports broad robustness to unfamiliar failures.

7 Limitations

Selective revision reduced incorrect assertions without improving completion over action verification. Its advantage over immediate revision depended on database task semantics. The targeted-check and decision-score comparisons showed no clear aggregate benefit.

The controllers start with correct memories, have a second working tool available, and use mostly hand-designed failure models. Only one observation-error scalar is fitted. Their cache predictions are approximate, they lose correlations between orders, and the unknown explanation cannot construct a new failure mechanism. Although the two environments have different actions and observation rules, both use the same short order structure. The tests cover only selected combined faults and one missing mechanism.

Sharing a planner controls part of the comparison, but it also makes every policy vulnerable to the same confirmation failure. Full history is implemented as a frequency rule, not an unrestricted use of past observations. The verification baselines are limited adaptations with a small tuning search. Equal budget caps do not mean equal calls, runtime, memory use, or evidence. No equal-token language-model comparison was run. The intervals describe variation in this generator across 40 base units; zero observed successes do not establish impossibility.

A next experiment would compare memory policies under different confirmation rules, including a rule that can stop relying on an unreliable channel. It would need a fresh holdout, the same public evidence, comparable budgets, and measurements of false acceptance as well as completion. Accepting unfinished work would not be an improvement. A later language-model pilot could test action selection and memory updates, starting with competence in no-change conditions. These experiments are proposed, not completed; the plan is in `docs/next_experiment_plan.md`.

8 Related work

Model-based diagnosis compares possible faulty components as explanations for observations [Reiter, 1987]. Active model discrimination chooses actions to distinguish tool, sensor, and compound faults, including situations where the true model is missing [Ni et al., 2026]. The fault categories and use of diagnostic actions in this study are established ideas.

Partially observable control formalizes decisions made without direct access to the full state [Kaelbling et al., 1998]. Bayesian experimental design selects informative checks [Foster et al., 2021], and dual control recognizes that actions can both alter a system and reveal information [Klenske and Hennig, 2016]. Change-point inference asks when a process changed [Adams and MacKay, 2007]; it does not alone say which memory field to revise. Selective prediction evaluates error together with how often a system chooses to answer [Geifman and El-Yaniv, 2017]. Model-misspecification results caution that normalized weights need not be reliable probabilities [Grünwald and van Ommen, 2017].

Closest to this study, BeliefWeaver combines belief updates with world modeling and checkpoint rollback [Wu et al., 2026]. Dependency-Guided Rollback Repair starts from diagnosed faulty memories and repairs their consequences [Wang et al., 2026b]. EnvProbe verifies task-relevant fields; inspected code supplies gold diagnostic probes despite delayed ordinary observations [Song and Cai, 2026]. AMV-Guard already considers action-conditioned verification, noisy evidence, value of information, and abstention, with compound and correlated cases in inspected development material [Wang et al., 2026a]. The symbolic adaptations here do not measure those full systems’ performance.

CounterMem uses counterfactual model checks with copying/reset and frozen-memory evaluation [Pu et al., 2026]. VISTA studies exploration and self-improvement in simulated embodied tasks [Zhang et al., 2026]. The retention position paper of Anand et al. [2026] distinguishes preserving unchanged knowledge from delaying needed updates. This study compares memory updates under fallible observations, persistent state, different completion rules, and explicit interaction costs. The negative results identify limits of this implementation; they do not establish priority for diagnostic action selection.

The literature review covers sources checked through 28 October 2026. Source versions and publication-status checks are recorded in `docs/novelty_audit.md`; one acceptance is author-reported, and one potentially relevant OpenReview source was inaccessible. These limits prevent a comprehensive priority claim.

9 Conclusion

Alice reduced incorrect memory edits but did not outperform action verification on task completion. Its lower edit count also came with lower supported retention. The shared planner failed under persistent stale observations even when Alice identified an unreliable channel. A next experiment should test confirmation rules that can use this diagnosis, while measuring false acceptance alongside completion.

References

- Ryan Prescott Adams and David J. C. MacKay. Bayesian online changepoint detection, 2007. URL <https://arxiv.org/abs/0710.3742v1>. Version 1, 19 October 2007.
- Nishit Anand, Ramani Duraiswami, and Dinesh Manocha. What should world models forget? stratified retention for continual adaptation, 2026. URL <https://arxiv.org/abs/2610.03713v1>. Version 1, 2 October 2026. Author-reported acceptance to NeurIPS 2026 Continual World Models Workshop.
- Adam Foster, Desi R. Ivanova, Ilyas Malik, and Tom Rainforth. Deep adaptive design: Amortizing sequential Bayesian experimental design. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3384–3395. PMLR, 2021. URL <https://proceedings.mlr.press/v139/foster21a.html>.
- Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL <https://proceedings.neurips.cc/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html>.
- Peter Grünwald and Thijs van Ommen. Inconsistency of Bayesian inference for misspecified linear models, and a proposal for repairing it. *Bayesian Analysis*, 12(4):1069–1103, 2017. doi: 10.1214/17-BA1085. URL <https://ir.cwi.nl/pub/27108>.
- Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X. URL <https://people.csail.mit.edu/lpk/papers/aij98-pomdp.pdf>.
- Edgar D. Klenke and Philipp Hennig. Dual control for approximate Bayesian reinforcement learning. *Journal of Machine Learning Research*, 17(127):1–30, 2016. URL <https://jmlr.org/papers/v17/15-162.html>.
- Xinpei Ni, Melkior Ornik, Glen Chou, and Samuel Coogan. Safe, real-time active model discrimination and fault diagnosis for nonlinear systems via differentiable reachability, 2026. URL <https://arxiv.org/abs/2606.19590v1>. Version 1, 17 June 2026. Preprint.
- Hongji Pu, Ruixiang Tang, and Yongfeng Zhang. COUNTERMEM: World-model verified counter-factual memory for language agents, 2026. URL <https://arxiv.org/abs/2609.31874v1>. Version 1, 25 September 2026. Preprint.
- Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987. doi: 10.1016/0004-3702(87)90062-2. URL <https://www.sciencedirect.com/science/article/pii/0004370287900622>.
- Xinyuan Song and Zekun Cai. Ask the world before acting: Environment probing for calibrating agent world models, 2026. URL <https://arxiv.org/abs/2606.31422v3>. Preprint, version 3, 20 October 2026; first submitted 30 June 2026. Under review. Verified 28 October 2026.
- Yiqi Wang, Yifei Wang, Jiaqi Zhang, Zhangkai Wu, Yiqun Duan, Taotao Cai, and Mingkai Zheng. When is a memory safe enough to act on? action-conditioned verification for long-lived agents, 2026a. URL <https://arxiv.org/abs/2610.31446v1>. Preprint, version 1, 22 October 2026. Verified 28 October 2026.
- Yiqi Wang, Caili Yu, Jiaqi Zhang, Yiqun Duan, Mingkai Zheng, Zhangkai Wu, Kaize Shi, and Taotao Cai. From faulty memories to corrected actions: Dependency-guided rollback repair for memory-augmented agents, 2026b. URL <https://arxiv.org/abs/2608.10502v2>. Preprint, version 2, 22 October 2026; first submitted 11 August 2026. Verified 28 October 2026.
- Xiyang Wu, Zhichao Liu, Shengxin Zhang, Zongxia Li, Ming Li, Tianyi Zhou, and Dinesh Manocha. Failure recovery via belief updates and world modeling for long-horizon agents, 2026. URL <https://arxiv.org/abs/2610.18082v1>. Preprint, version 1, 13 October 2026. Verified 28 October 2026.
- Zhang Zhang, Shengjun Zhang, Simin Huang, Min Chen, Chong Xia, Xin Fei, and Yueqi Duan. VISTA: Active exploration for self-improvement in embodied agents, 2026. URL <https://arxiv.org/abs/2610.35051v1>. Version 1, 26 October 2026. Preprint.

A Formal model and bounded diagnostic checks

At cumulative interaction-cost time t , the hidden state is

$$s_t = (x_t, w_t, c_t, o_t, z_t, j_t).$$

The state records backend values x_t , credential rules w_t , tool health c_t , channel freshness o_t , cached answers z_t , and public order index j_t . Here c_t is tool state, whereas c in Table 1 is the read price. If an action crosses a scheduled change time, the change occurs before that action takes effect. The simulator selects a live or cached answer and then applies bit-flip noise. Noise depends on the scenario seed, cost time, and action fields; policies can still encounter different states and times.

An agent’s history is $h_t = (a_1, y_1, \dots, a_t, y_t)$, with actions a_i and responses y_i ; persistent memory is M_t . A response contains a binary value, or null for advance, together with cost time and order index. The simulator controls hidden state and change schedules. The evaluator reads hidden variables for scoring, but agents receive no backend access, true parameters, success flag, or event label. Identifiers do not reveal failure causes. This Python interface was audited for information leakage; it is not a security sandbox against malicious introspection.

A separate diagnostic test starts with the same visible failure under a world, tool, or observation fault. In its database cases, the initial read checks durable completion and the observation fault freezes both channels. Ordinary database episodes need not share this exact failure signature. The test enumerates 16 candidate actions, and a known-model decision tree considers every binary response for up to two further actions. Replay on the evaluator side constructs the response distributions. The solver receives those distributions, not the realized hidden state.

With equally likely causes, no diagnostic penalty, and compulsory classification, optimal expected accuracy is $1/3$, $2/3$, and 1 after zero, one, and two further actions. The noiseless two-action strategy costs $7/3$ units on average, excluding the initial two-unit apply. At known independent error rates 0.05 and 0.15 , two-action accuracy falls to 0.9183 and 0.7650 . Greedy information gain matches optimal decision risk in all 108 paired configurations (216 policy rows). With per-unit penalty 0.2 and abstention loss 0.4 , both abstain. These checks establish bounded diagnostic feasibility and cost sensitivity, not an optimal deployment policy or upper bound on episode success.

B Check selection

Let q denote the current weights over explanations. For candidate check a , Alice predicts possible answers y using its approximate observation model $\hat{P}$. Updating with that answer would give weights $q_{a,y}$. Alice scores the expected reduction in loss, less the check’s cost:

$$G(a) = L(q) - \mathbb{E}_{y \sim \hat{P}(\cdot | a, q)} L(q_{a,y}) - \lambda \text{cost}(a).$$

The score uses the controller’s model, not evaluator truth. Its loss combines three memory decisions and the best predicted physical action risk:

$$L(q) = f(p_1) + f(p_2) + f(p_3) + \min_u \hat{P}(\text{physical failure} \mid u, q).$$

Here p_1, p_2, p_3 give the total weight on each fault type: world, tool, and observation. Half the unknown weight is allocated to each binary value. The function $f(p) = \min\{p, 1 - p, 0.2\}$ compares the expected cost of choosing either value with the fixed cost of deferring. The action u ranges over apply routes. The tuned cost penalty is $\lambda = 0.06$. This score approximates the value of a single check; it neither accounts for all later decisions nor solves the planner’s confirmation problem.

A check requires a positive score, enough budget, and room under the limit of three probes per order. The implementation counts mandatory reads toward that limit. Action verification scores only physical failure risk. The information-gain variant scores expected entropy reduction instead of loss reduction. Random-check variants use the same decision to check but select a random candidate; their evidence and realized costs can consequently differ.

The remaining parameters are hand-set: response-level hazard 0.04 , secondary correlation 0.15 , unknown likelihood 0.5 , and unknown prior 0.05 . The filter approximates a stale answer as zero. To avoid treating repeated reports as independent, it raises the likelihood of an identical same-channel read or inspection to 0.25 . The provenance ablation removes this discount. Although memory entries reserve dependency fields, those fields do not form a populated or learned graph; the ablation does not test general graph repair.

Selective revision requires marginal support of 0.80 . Unknown weight at least 0.30 marks the three implicated fields uncertain. A controlled variant collapses the joint weights after every response. Another removes the unknown component. A separately labeled parameter-informed reference receives true scalar noise and correlation but uses the same approximate filter and planner; it is distinct from the exact finite diagnostic check in Appendix A.

A shared predictor logs probabilities before responses and reports fault attributions for every policy. Some policies do not use these weights to select actions. Their logging scores therefore describe the shared predictor, not diagnostic reasoning by each policy. Reported costs include this instrumentation.

C Protocol, aggregation, and additional measurements

Seven main controllers each received two predeclared configurations on 128 development scenarios. Selection prioritized success, then correct-order fraction, then lower cost, with a fixed tie break. Variants inherited Alice’s settings. A further 7,680-episode development run estimated variability and runtime. Training used cyclic orders, development rotated/alternating orders, and final evaluation adjacent paired visits with shuffled blocks. Final noise values were 0.05 and 0.10; initial changes occurred at cost time 11 or 12. Split details are in the frozen configuration.

The largest pilot standard deviation of paired aggregate success differences was 0.07855 across eight base units. Planning instead used a conservative standard-deviation floor of 0.20 and a nominal interval half-width target of 0.08, requiring 25 units; the evaluation used 40. Sample size was not chosen from the final significance of Alice’s results.

The first local freeze, interrupted run, and correction are retained. The corrected freeze is timestamped 28 October 2026 at 21:45:01 UTC. The amendment preserves training/development trajectories, selected controllers, budgets, metrics, and sample size; only final onset variation and seed namespace changed. The 8,714 interrupted records are excluded from scientific estimates but included in resource accounting. The full audit is `docs/protocol_amendment_v2.md`.

The five planned primary success comparisons use 20,000 cluster sign flips with Holm correction. Their p -values rely on exchangeability or symmetry of paired differences; the deterministic policies were not randomly assigned as treatments. The bootstrap seed is 20261028. Reusing bootstrap samples preserves pairing, but the percentile intervals remain individual rather than simultaneous intervals. Secondary comparisons and later exploratory analyses are labeled separately. Exceptions, timeouts, and invalid actions count as failed episodes under the fixed rule, without retries to improve outcomes.

C.1 Metric aggregation

For condition k , let H_k count incorrect accepted value changes across its 40 episodes and A_k count all accepted value changes. If K conditions have positive A_k , the reported rate is

$$R = \text{mean}_{k: A_k > 0} (H_k / A_k).$$

All $K = 48$ primary conditions contribute for Alice, action verification, and immediate revision. Pooling across conditions before division weights conditions by their edit counts instead. For Alice that pooled rate is $591/2656 = 22.3\%$, whereas equal-condition averaging gives 31.1% . The difference comes from weighting conditions rather than edits. The full-history controller makes no value-changing edits and has no defined rate.

Supported retention also averages pooled rates equally across conditions, using the field/action opportunities defined in the main text. Edit coverage counts episodes instead. Reaffirming a value is recorded separately from changing it. The implementation’s name “harmful edit” means a false assertion, not a demonstrated causal effect on task performance.

The recovery window excludes an acceptance at the next change and ends at first progress or censoring. Alice’s mean observed window length is 2.63 cost units, including censored windows; it is not a time to sustained recovery. Averaging episode censoring fractions equally gives 5.2% . Pooling windows instead gives 158 of 2,240, or 7.1% . These use different weights. Censoring may depend on the difficulty of recovery, which limits interpretation.

Attribution is scored using the latest report available eight cost units after a change. The report must identify the exact world/tool/observation combination for the current target; unknown reports are separate. Alice reports an attribution in 29.5% of primary cases and is correct in 46.3% of those cases. Action verification has 37.2% coverage and 66.9% accuracy. These are target-specific logging measures, not global diagnosis scores. The observation Brier score is the mean squared error of binary-response probabilities recorded before each response. Alice scores 0.169 in the main population and 0.312 with the missing mechanism. This does not establish calibration of fault weights, and the policies observe different selected evidence.

D Complete comparisons and controlled variants

Controller	Success (%)	95% CI	Orders (%)
History frequency	40.5	[38.5, 42.2]	66.3
Immediate revision	44.6	[42.6, 46.4]	67.8
Recency revision	44.6	[42.6, 46.4]	67.8
Periodic probing	39.7	[38.1, 41.2]	65.0
Field verification	22.1	[20.8, 23.4]	54.6
Action verification	50.8	[48.8, 52.6]	70.4
Alice	46.6	[44.8, 48.1]	68.9

Table 3: All seven main controllers: 1,920 episodes per controller, 48 equally weighted conditions. Success requires all 24 orders; orders is the correctly accepted fraction of those 24. Each interval is a pointwise 95% cluster-bootstrap interval from the same 40 base units.

Alice minus	Difference (pp)	95% CI	Holm p
Action verification	-4.2	[-5.8, -2.7]	< 0.001
Field verification	24.4	[22.9, 25.9]	< 0.001
History frequency	6.0	[4.5, 7.7]	< 0.001
Immediate revision	2.0	[0.6, 3.4]	0.026
Recency revision	2.0	[0.6, 3.4]	0.026

Table 4: Five planned primary success contrasts. Differences and intervals are in percentage points. Intervals are pointwise paired 95%; Holm-adjusted sign-flip p -values use the stated assumptions and do not change interval coverage.

Factor contrast	Success (pp), 95% CI	False edits, 95% CI
Targeting	0.10 [-0.55, 0.73]	-0.21 [-0.28, -0.12]
Selection	-2.55 [-3.80, -1.33]	-3.07 [-3.25, -2.87]
Interaction	0.31 [-0.99, 1.56]	0.49 [0.37, 0.61]

Table 5: Planned secondary factorial contrasts, with pointwise paired 95% intervals. Targeting averages targeted-minus-random differences across the two revision rules; selection averages selective-minus-joint-commitment differences across the two checking rules; interaction is the difference of the two targeting effects. No extra confirmatory tests are claimed.

Setting: regime	Immediate	Action verify	Alice	Entropy
Gates: control	90.0	92.5	83.1	92.5
Gates: world	80.6	75.6	73.1	71.2
Gates: capability	83.1	83.8	75.0	71.9
Gates: observation	0.0	0.0	0.0	0.0
Gates: repeated	60.0	56.9	51.9	47.5
Gates: correlated	0.0	0.0	0.0	0.0
Gates: compound	31.9	31.2	26.9	27.5
Gates: unknown	78.8	71.2	31.2	23.1
Ledger: control	86.2	91.2	91.9	91.2
Ledger: world	82.5	81.9	73.1	72.5
Ledger: capability	11.9	78.8	74.4	70.6
Ledger: observation	0.0	0.0	0.0	0.0
Ledger: repeated	40.6	48.8	36.2	40.6
Ledger: correlated	0.0	0.0	0.0	0.0
Ledger: compound	16.9	38.8	32.5	30.6
Ledger: unknown	76.9	73.8	39.4	19.4

Table 6: Success percentages by environment and change condition, equally averaged over four budget/price cells: 160 episodes per row/controller. Compound and unknown rows are separate robustness populations. “Unknown” means the execution failure missing from the explanation list; “entropy” is the information-gain variant. These descriptive subgroup means are not significance tests.

Controller	Success (%)	Orders (%)	False edits	Units
Alice, entropy gain	46.5	68.7	0.42	164.4
Non-targeted, selective	46.3	68.7	0.27	166.0
Targeted, MAP	49.0	69.9	3.13	161.9
Non-targeted, MAP	49.0	69.6	3.58	164.0
Premature commitment	33.4	65.4	0.61	174.3
No unknown component	47.5	69.0	0.44	164.1
No source discount	46.6	68.9	0.31	164.3
Parameter-informed reference	45.4	68.9	0.12	164.6

Table 7: Main-population outcomes for controlled variants. “False edits” means incorrect assertions per episode, and units are mean interaction cost. The final row is a parameter-informed approximate reference, not an optimal policy. Exact paired variant intervals remain in the saved analysis.

D.1 Field-verification cost

The selected field-verification threshold of 1.5 causes an inspection on nearly every order after the first. At read price 3, a no-error run with 23 inspections already needs at least $24(2 + 3 + 1) + 23(6) = 282$ units, exceeding both budgets. This explains zero success in the high-price no-change conditions. The search over two configurations did not produce a well-tuned policy for every price. These results concern this adaptation, not the published EnvProbe system.

Without the unknown explanation, main success is 47.5%. Removing the repeated-source discount changes success by one episode in 1,920. Giving the approximate model the true scalar parameters yields 45.4% success, so correct parameter values alone do not fix its structural weaknesses. With the missing mechanism, removing unknown gives 33.4% rather than Alice’s 35.3%; this descriptive difference does not show that Alice discovered the mechanism. Its reported attributions are all wrong in that condition, at 5.0% coverage.

E Memory coverage and measured costs

Controller	Edits	False	Collateral	Edit cov.	Retention
History frequency	0.00	0.00	0.00	0.0	100.0
Immediate revision	28.81	14.63	14.62	98.8	97.3
Recency revision	25.82	13.14	13.14	98.8	97.5
Periodic probing	29.49	15.02	15.00	98.8	97.2
Field verification	28.02	14.31	14.30	99.7	97.4
Action verification	7.59	3.38	3.34	97.9	99.5
Alice	1.38	0.31	0.30	86.2	90.5

Table 8: Main-population memory outcomes. Edits, false edits, and collateral edits are means per episode, retaining zero-edit episodes. Coverage and supported retention are percentages with the different denominators defined in the text. Full history’s perfect opportunity-retention score excludes changed truths and does not establish adaptation.

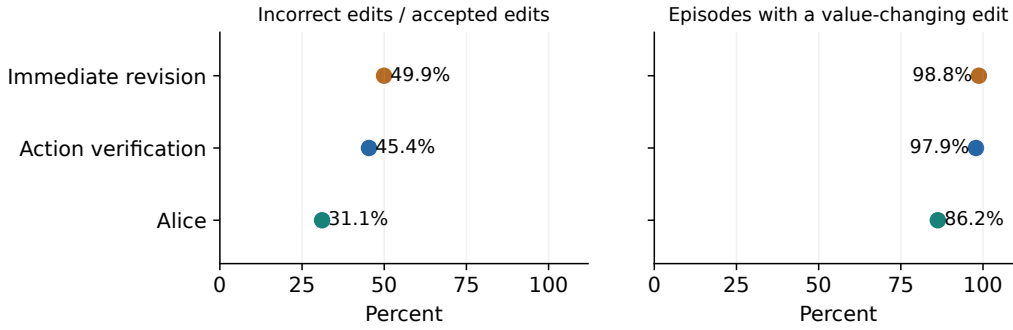


Figure 3: Edit error and coverage for the three controllers in Table 2, using the main population. The left panel averages each condition’s incorrect-to-accepted edit ratio equally. The right shows episodes with at least one changed value. Uncertainty bars are omitted. Neither measurement alone establishes reliable retention or task completion.

Controller	Units	Calls	Read/probe	Decision ms	Memory kB
History frequency	168.5	93.8	30.0	8.35	1.47
Immediate revision	165.2	91.7	25.3	3.66	1.54
Recency revision	165.2	91.7	25.3	3.61	1.54
Periodic probing	171.9	91.6	28.0	3.65	1.54
Field verification	187.4	93.0	37.7	3.69	1.56
Action verification	161.1	89.9	25.8	12.39	2.11
Alice	164.4	91.2	28.0	11.15	2.15

Table 9: Mean main-population costs per episode. Read/probe includes mandatory reads. Decision time includes selection, response processing, and shared predictor logging. Memory is peak serialized structured-entry size averaged across episodes, in decimal kB; it excludes raw-history storage and process RAM. Timings are implementation measurements, not portable performance guarantees.

F Reproducibility

The repository includes the environments, controllers, pinned dependencies, 366 unit/integration tests, information-boundary checks, frozen source/manifests, separate agent/evaluator logs, analysis, and manuscript assets. It retains all 38,400 corrected final records. Replaying six frozen cases in a clean macOS ARM64 environment with Python 3.12 reproduced identical agent traces and non-timing metrics. This was neither an independent full-study rerun nor a cross-platform check.

Final episode execution used 792.154 CPU seconds and 857.532 summed episode wall seconds, excluding setup, serialization, and analysis. Interrupted and development runs are documented in `docs/compute_report.md`. The experiment used zero language-model calls, zero language-model tokens, and zero additional external spending.

The reproduction guide, `docs/reproduction.md`, covers installation, tests, smoke runs, six-case replay, the 38,400-episode study, and reanalysis. Source hashes and timestamps are in `runs/frozen-v2`; `docs/claim_evidence.csv` links claims to evidence. Build instructions and manuscript assets are included. Transfer to language-model agents and deployed systems remains untested.