Before the Fall: Delta Minimal Failing Prefixes for Local Tool-Use Agent Failures

Daoyuan Li [1]

Abstract

Final success rates conflate qualitatively different tool-use agent failures: a model may fail from the initial interface, enter a mid-trajectory cascade, behave unstably under replay, or make an irreversible state change. We introduce Δ Minimal Failing Prefixes (Δ-MFP), a counterfactual-replay diagnostic that compares the failure probability p_k from a trajectory prefix to the failure probability p_0 from the initial state, classifying each failed trace as prefix-0 ($p_0 \geq p_f$), a nontrivial cascade (some prefix k with $p_k \geq p_f$ and $p_k - p_0 \geq \delta$), or unstable/no-Δ. We build a deterministic local tool-use suite of 120 calendar, refund, file/email, and inventory tasks and run three probes entirely on single-GPU local inference with Qwen2.5-7B and Llama-3.1-8B under four tool interfaces: natural failures, persistent fault injection as a positive control, and a soft fault-injection probe that perturbs evidence, memory, or arguments without a state violation. On 25 natural failures, 13/25 are nontrivial Δ-MFPs, 5/25 prefix-0, and 7/25 unstable, and the 40 persistent injections all localize at the injected prefix (a positive control for the replay engine). A full $N = 5$ replay audit of all 50 soft traces classifies 7/50 as nontrivial Δ-MFPs, 21/50 as unstable/no-Δ, and 22/50 as prefix-0; although this aggregate profile nearly matches the $N = 2$ one, only 1 of the 7 $N = 2$ nontrivial localizations survives at $N = 5$, so Δ-MFP must be reported with replay uncertainty and unstable/no-Δ is a substantive outcome rather than a discarded error category. A repair probe over retry, reflection, fault-specific gates, and a rule-based router ($N = 3$ replay, $n \leq 5$ per cell, Wilson 95% CIs) is reported as a diagnostic rather than a ranking; the package is a reproducible failure-regime profile for local tool-use agents, not a claim that lightweight gates solve repair.

[1]University of Minnesota Twin Cities, Minneapolis, Minnesota, USA. Correspondence to: Daoyuan Li <li002504@umn.edu>.

Accepted at the ICML 2026 Workshop on Failure Modes in Agentic AI (FAGEN), Seoul, South Korea. Copyright 2026 by the author(s).

1. Introduction

Tool-use agents fail in ways that are hard to understand from final success alone. A failed calendar assistant may never have understood the JSON tool interface; it may have understood the interface but committed to the wrong event identifier; it may alternate between valid and invalid continuations depending on stochastic decoding; or it may issue an irreversible refund before checking policy. These cases have different implications for benchmark design and repair. A prefix-0 interface failure calls for scaffolding or model capability improvements. A mid-trajectory cascade calls for trace-level diagnosis and targeted intervention. An unstable failure calls for uncertainty reporting. An irreversible state change calls for precondition guards or rollback, not merely a better final answer.

Most agent evaluations still compress these distinctions into end-to-end success. This is understandable: final success is easy to score and is the quantity users often care about. But it is incomplete as a failure-analysis signal. In a stateful tool environment, the relevant question is often not only whether the trajectory failed, but *when failure became reproducible*. If replaying from the initial state already fails with high probability, then a later "root cause" is suspect: the model was already in a failure regime before that step. If replaying from a later prefix fails much more often than replaying from the beginning, then that prefix is a reproducible trigger candidate. If no prefix reliably fails, the trajectory may be an unstable sample rather than a stable cascade.

We study this problem under a deliberately local-compute constraint: all model experiments use local inference on RTX 4080-class hardware. This constraint matters for agent failure analysis because many practical tool agents, research prototypes, and privacy-sensitive deployments run on local 7B-14B models rather than closed frontier APIs. A protocol that requires hours of API budget per trace cannot be the basis of a workshop-shareable failure-analysis methodology. Local models also expose an additional confound: strict tool interfaces can dominate failure analysis. A JSON parsing failure at step zero and a genuine state-contamination cascade should not be reported as the same kind of tool-use failure. Reporting prefix-0 rate as a first-class metric, alongside cascade rate and unstable rate, lets practitioners decide whether to invest in interface scaffolding, decoding stability,

or trace-level repair.

We propose Δ Minimal Failing Prefixes (Δ-MFP), a counterfactual replay diagnostic that compares the estimated failure probability from prefix k to the failure probability from the initial state. The ordinary Minimal Failing Prefix asks for the earliest prefix whose replay fails with probability at least p_f. Δ-MFP adds a baseline correction: a nontrivial prefix must both exceed p_f and increase failure probability by at least δ relative to prefix 0. This small modification turns MFP from a generic replay statistic into a failure-regime decomposition: prefix-0, nontrivial cascade, unstable/no-Δ, and irreversible or high-cost repair.

Our empirical core has four parts: (i) a deterministic local tool-use testbed with 120 tasks across calendar, refund/order, file/email, and inventory workflows, evaluated with local Qwen2.5-7B, Llama-3.1-8B, and a small Qwen2.5-14B probe under four interfaces (raw JSON, JSON+Repair, schema-guided, tool compiler); (ii) natural-failure Δ-MFP classification on calibrated model-interface-task cells; (iii) persistent fault injection as a positive control for the replay machinery; and (iv) a soft fault-injection probe that perturbs evidence, memory, or arguments without committing a state violation, so localization is harder and more realistic. We then evaluate replay repairs (Clarify, Precondition, Evidence, Rollback, Boundary), compare a rule-based predicted router using only prefix-visible features against an oracle upper bound, and report success rates separately for persistent and soft.

The results are modest but informative. In the natural set 13/25 failed traces have nontrivial Δ-MFPs, 5/25 are prefix-0, and 7/25 are unstable. In the persistent positive control all 40 traces are nontrivial at the injected prefix. The soft probe is harder: using a full $N = 5$ replay audit of all 50 soft traces, 7/50 are nontrivial Δ-MFPs, 21/50 are unstable/no-Δ, and 22/50 are prefix-0. This aggregate profile is close to the $N = 2$ profile (7/23/20), but the per-trace classification is not: only 1 of the 7 $N = 2$ nontrivial localizations remains nontrivial at $N = 5$, while 6 traces that were unstable or prefix-0 become nontrivial, so disagreements move in both directions. Low-replay Δ-MFP can therefore both over- and under-attribute failures to a perturbation prefix; we report Δ-MFP with replay uncertainty and treat unstable / no-Δ as a substantive outcome rather than an error category. Repair is a diagnostic probe ($N = 3$ replay, $n \leq 5$ persistent and $n = 4$ soft per method) that we report with Wilson 95% CIs and explicitly do not rank; the rule-based router often selects the same repair as oracle on the persistent positive-control set but cannot be calibrated on soft.

Our contributions are:

1. We formalize Δ-MFP as a replay-based failure-regime decomposition for stateful tool-use agents.
2. We release a capability-calibrated local tool-use suite, four interface conditions, and persistent plus soft fault-injection probes that separate interface brittleness, mid-trajectory cascades, and quiet evidence perturbations.
3. We report local LLM experiments on natural failures, persistent and soft fault injection (with a full $N = 5$ replay audit of all 50 soft traces), verified repairs at $N = 3$, and a fully rule-based predicted router under a single-GPU budget.
4. We argue that MFP analysis should be reported as a profile rather than a single number: prefix-0 rate, nontrivial cascade rate, unstable/no-Δ rate, localization distance, and repairability by fault type and class.

2. Related Work

Tool-use and function-calling evaluation. Early benchmarks (API-Bank, ToolLLM, Gorilla, ToolAlpaca) score tool selection, API arguments, and hallucinated calls (Li et al., 2023; Qin et al., 2024; Patil et al., 2024; Tang et al., 2023). Newer benchmarks move closer to deployed agents: ToolSandbox (Lu et al., 2024), τ-bench (Yao et al., 2025), BFCL (Patil et al., 2025), and MCP-Bench (Wang et al., 2025). These benchmarks score final task success or function-call correctness rather than estimating the earliest prefix from which failure reproducibly recurs or evaluating repairs at that prefix.

Trace-level failure diagnosis. Several recent systems treat failed trajectories as diagnostic objects: AgentDebug (Zhu et al., 2025), AgentRx (Barke et al., 2026), AGENTSCOPE (Bi et al., 2026), AgenTracer (Zhang et al., 2025), and DoVer (Ma et al., 2026). Our work is complementary but asks a different operational question: after correcting for the initial failure probability, what is the shortest trajectory prefix whose replay makes failure substantially more likely, and which lightweight repair changes that replay outcome?

Long-horizon environments and safety failures. WebArena, VisualWebArena, OSWorld, and AppWorld demonstrate long-horizon failures in realistic environments (Zhou et al., 2024; Koh et al., 2024; Xie et al., 2024; Trivedi et al., 2024). Safety benchmarks (AgentDojo, AgentHarm, AgentHazard, Agent-SafetyBench) show that tool output and accumulated context create failure modes beyond single-turn robustness (Debenedetti et al., 2024; Andriushchenko et al., 2025; Feng et al., 2026; Zhang et al., 2024). We include boundary-trigger failures in our suite while keeping the central experiment focused on reproducible diagnostics.

Repair and reflection. ReAct-style agents (Yao et al., 2023) and Reflexion (Shinn et al., 2023) provide our retry /

Work	Final success	Trace diagnostics	Counterfactual replay	Minimal prefix	Verified repair	Stateful tools
ToolSandbox	yes	milestones	no	no	no	yes
τ-bench	yes	partial	full episode	no	no	yes
BFCL V4	yes	partial	no	no	no	partial
AgentDebug	yes	yes	no	no	feedback	mixed
AgentRx	yes	yes	no	no	no	yes
AgenTracer	yes	yes	yes	no	feedback	multi-agent
DoVer	yes	yes	interventions	no	yes	multi-agent
Ours	yes	yes	yes	yes	yes	yes

Table 1. Comparison to related benchmarks and diagnostic methods. Our contribution is not another broad benchmark, but a replay protocol that estimates when a failed trajectory enters a reproducible failure basin and tests repairs at that point.

reflection baselines. Recent repair-oriented work treats diagnosis and repair as first-class concerns (Mulian et al., 2026; Gong et al., 2026). Our repairs are deliberately lightweight inference-time interventions (clarification, state check, evidence grounding, rollback, untrusted-data guarding) applied at replay-identified prefixes; the method is a replay protocol, not a new broad benchmark.

3. Failure Regimes and Δ-MFP

Trajectories and state. We consider a stochastic tool-use policy π_θ interacting with a deterministic stateful environment. A trajectory is

$$\tau = (s_0, o_0, a_0, r_0, s_1, \ldots, o_T, a_T, r_T, s_{T+1}),$$

where s_t is the simulator state, o_t is the observation shown to the model, a_t is either a tool call, clarification, or final answer, and r_t is the tool result or environment response. The trace logger records tool arguments, tool returns, memory writes, state diffs, parser status, and final evaluation. A prefix $\tau_{\leq k}$ denotes the trace and simulator snapshot after k logged actions, with $k = 0$ meaning before any agent action.

Failure. Each task provides required effects, forbidden effects, and state invariants. A trajectory fails if it misses a required effect, produces a forbidden side effect, violates a tool-state invariant, violates a policy constraint, or terminates without a valid final state. We use rule-based evaluators for ground truth; LLM judges are not used as success criteria.

Ordinary MFP. For a failed trajectory and prefix k, let

$$p_k = \Pr_{\pi_\theta, \xi}\left[\text{Fail} \mid s_k, h_k\right],$$

where replay restores simulator state s_k and trace history h_k, then samples a fresh continuation using replay seed ξ. Given replay count N and threshold p_f, an ordinary Minimal Failing Prefix is the smallest k such that $\hat{p}_k \geq p_f$.

Failed local agent trace, four regimes:	
Prefix-0	$\hat{p}_0 \geq p_f$; failure already reproducible from the initial state.
Nontrivial Δ-MFP	smallest $k > 0$ with $\hat{p}_k \geq p_f$ and $\hat{p}_k - \hat{p}_0 \geq \delta$.
Unstable / no-Δ	no prefix reaches p_f, or some prefix exceeds p_f without a meaningful increase over $\hat{p}_0$.
Irreversible / costly	a Δ-MFP exists but lightweight repairs cannot recover the contaminated state.

Figure 1. Failure-regime decomposition for a failed local tool-use trace. Each row defines one regime in terms of the replay-estimated failure probabilities $\hat{p}_0$ and $\hat{p}_k$, the regime threshold p_f, and the increase threshold δ.

Δ-MFP. Ordinary MFP can collapse to prefix 0 when the model or interface is already unreliable from the initial state. We therefore define a nontrivial Δ-MFP as the smallest prefix $k > 0$ satisfying

$$\hat{p}_k \geq p_f \quad \text{and} \quad \hat{p}_k - \hat{p}_0 \geq \delta.$$

Unless otherwise noted, we use $p_f = 0.6$ and $\delta = 0.3$. Natural-failure replay uses $N = 3$ in our local-compute run; persistent fault-injection localization uses $N = 2$ because state-violation prefixes are deterministic under the evaluator; the soft set is audited at $N = 5$. These values are smaller than ideal benchmark-scale replay, but every retained trace reports its replay regime rather than silently dropping ambiguous cases.

Failure regimes. Figure 1 summarizes the regime decomposition. A *prefix-0 failure* has $\hat{p}_0 \geq p_f$ and no later prefix produces a meaningful increase. A *nontrivial cascade* has a Δ-MFP. An *unstable* trace has no prefix reaching p_f under replay. A *no-Δ* trace has some failing prefix but not enough increase over $\hat{p}_0$. An *irreversible* or high-cost repair case may have a clear Δ-MFP but fail all lightweight repairs because the state has already been contaminated.

4. Method

Replay protocol. For each failed trace, the replay engine restores a saved simulator snapshot and trace history, then samples fresh continuations from the same local model with distinct seeds. Prefix 0 restores the initial state before any agent action. Prefix k restores the state after the kth logged action, including any memory writes, tool effects, and injected state faults. Replays never copy the original failed suffix. If a snapshot already contains an unrepaired persistent state violation, the evaluator deterministically marks the replay as failed; this is a runtime optimization equivalent to executing any continuation because the violation remains in the final state.

Prefix search. Natural-failure analysis scans all saved prefixes for the retained failed traces. Fault-injection analysis uses the targeted prefix set $\{0, k_f\}$, where k_f is the injected-fault prefix, because injected persistent faults are positive controls: the scientific question is whether the fault produces a nontrivial basin relative to prefix 0, not whether a binary search can rediscover an arbitrary hidden step. For every trace we report prefix-0, nontrivial Δ-MFP, unstable, or no-Δ status. We do not remove prefix-0 or unstable traces from denominators in the natural-failure analysis.

Local tool interfaces. The same model is evaluated under four interfaces. **Raw JSON** requires exact tool-call JSON. **JSON+ParseRepair** permits one syntax-only repair retry when the output is invalid JSON. **Schema-Guided** repeats tool schemas, required fields, current state summaries, and completion criteria at each step. **Tool-Call Compiler** lets the model write a natural-language intent and arguments, then uses a deterministic adapter to map the intent to the closest valid tool call. The compiler cannot invent task facts; if required arguments are missing, it emits a clarification action.

Fault injection. To test mid-trajectory localization in real local LLM loops, we collect successful or near-successful traces and inject persistent faults after an intermediate tool step. The implemented fault types include wrong identifier, stale memory, precondition skip, date ambiguity, missing field, and untrusted instruction. The retained main-table traces cover wrong identifier, stale memory, precondition skip, date ambiguity, and one missing-field case; untrusted-instruction traces were not numerous enough after filtering for the main table. Each injected fault is represented as a simulator-state violation until a compatible repair clears it, which makes the experiment a positive-control localization test rather than a natural-failure distribution estimate.

Repairs and routing. We evaluate both restart-style and prefix-level repairs. **Generic retry** restarts from prefix 0 without a special repair. **Prompted reflection** restarts with a short failure summary and a conservative precondition gate. **ClarifyGate** forces clarification when required arguments are missing. **PreconditionGate** blocks mutating tools until a required check has occurred. **EvidenceGate** patches action arguments from concrete tool-output fields. **RollbackRetry** restores the last clean state before the injected fault when possible. **BoundaryShield** treats external content as untrusted data. The **Predicted** router is rule-based: it scans the trajectory for prefix-visible markers (missing required field, identifier mismatch, ambiguous date, stale lookup, untrusted-source marker, precondition-skip marker, mutating-before-check) and selects a repair from a fixed ordered rule set. The exact rule order is documented alongside the released code. Predicted routing never sees the injected fault label. **Oracle** routing maps each injected fault type to its compatible repair and is reported only as an upper bound.

For step-budget fairness, restart-style methods (Retry, Reflection) are allowed up to the task's natural step budget; prefix-level repairs are capped lower because they only need to finish the remaining steps. The repair replay uses $N = 3$ on a single-GPU budget.

5. Local Tool-Use Testbed

Tasks. The local suite contains 120 deterministic tasks: four families (calendar, refund/order, file/email, inventory), three difficulty tiers (easy/medium/hard), and ten tasks per family-tier cell. Easy tasks require two or three explicit tool steps; medium tasks require four to six steps with one ambiguity or state dependency; hard tasks require six to ten steps with a memory dependency, precondition, policy constraint, or boundary-style issue. Each task specifies initial state, available tools, required effects, forbidden effects, state invariants, and fault-compatible fields.

Models and compute. All experiments use local Ollama inference. We successfully ran Qwen2.5-7B, Llama-3.1-8B, and a Qwen2.5-14B probe on RTX 4080-class hardware. Qwen2.5-7B is the main model because it gave the best throughput for repeated replay; Llama-3.1-8B and Qwen2.5-14B are used in calibration and runtime inventory. Recorded probe latencies were approximately 2.3s for Qwen2.5-7B, 5.9s for Llama-3.1-8B, and 6.6s for Qwen2.5-14B.

Calibration. Before MFP analysis, we run each model-interface-task cell for five seeds and classify the cell as too hard if pass rate is below 0.2, calibrated if pass rate is in $[0.2, 0.8]$, and too easy if pass rate exceeds 0.8. Table 2 reports 152 evaluated cells. Qwen2.5-7B under Raw JSON and JSON+Repair provides the largest number of calibrated cells; Tool compiler raises pass rate but makes many easy and medium tasks too easy for failure analysis.

Model	Interface	Tasks	Pass	Calibrated	Too hard	Too easy	Parse fail
Llama-3.1-8B	Tool compiler	16	0.500	0	8	8	0.000
Qwen2.5-14B	Tool compiler	16	0.844	1	2	13	0.000
Qwen2.5-7B	JSON+Repair	40	0.625	7	12	21	0.000
Qwen2.5-7B	Raw JSON	40	0.640	5	13	22	0.000
Qwen2.5-7B	Tool compiler	40	0.805	3	6	31	0.000

Table 2. Model/interface calibration summary.

Natural failures (regime estimator). We collect natural failed traces from calibrated cells, prioritizing Qwen2.5-7B across Raw JSON, JSON+Repair, and Tool compiler. The retained natural-failure set contains 25 failed traces. We replay saved prefixes and classify with $p_f = 0.6$, $\delta = 0.3$, and $N = 3$. The natural set is intrinsically rare in calibrated cells; we report it as a regime estimator rather than a benchmark.

Persistent fault injection (positive control). The persistent probe is an implementation positive control, not the main empirical claim. We inject persistent faults into successful or near-successful Qwen2.5-7B traces; once injected, the simulator state carries an explicit violation until a compatible repair clears it. After consistency filtering, we retain 40 failed traces spanning wrong-ID, stale-memory, precondition-skip, date-ambiguity, and missing-field. Replay tests $\{0, k_f\}$ with $N = 2$, $p_f = 0.6$, $\delta = 0.3$. The expected outcome is that all 40 traces produce nontrivial Δ-MFPs at the injected prefix; this verifies that snapshot restoration and repair-aware violation clearing are working, but it does not constitute strong empirical evidence about real failures.

Soft fault injection (realistic stress test). The soft probe is the harder, more realistic counterpart. We add six subtypes that perturb evidence, observation, memory, or arguments *without* committing a state violation: *soft wrong-id* (a distractor identifier replaces a tool-call argument and surfaces in the observation), *soft stale-memory* (an outdated value is written into agent memory and observation), *soft missing-field* (a required field is omitted from tool output and the next observation), *soft untrusted-instruction* (external content embeds an explicit injection prompt marked as untrusted), *soft date-ambiguity* (concrete date/time in arguments are replaced by ambiguous phrasing), and *soft precondition-ambiguity* (memory suggests a required check has already been satisfied when it has not). We re-use successful or near-successful trace snapshots as bases, apply the perturbation, and let the local model continue. Failure occurs only if the agent acts on the perturbation. We retain only attempts whose continuations actually fail and classify them with Δ-MFP using $p_f = 0.6$ and $\delta = 0.3$. After this filter, we retain 50 failed soft traces with subtype counts 17 date-ambiguity, 8 untrusted-instruction, 8 wrong-id, 7 missing-field, 5 precondition-ambiguity, 5 stale-memory. We re-classify all 50 at $N = 5$ replay as the camera-ready headline (the reviewed version audited only a stratified 27/50), and retain the $N = 2$ classification for comparison. The resulting localization for both classes is summarised below and in the released data; the soft block is the empirically informative one.

Repair evaluation. Repair is evaluated on persistent and soft repair subsets reported independently (no cross-class fallback). We compare *None*, *Retry*, *Reflection*, *ClarifyGate*, *PreconditionGate*, *EvidenceGate*, *RollbackRetry*, *BoundaryShield*, the rule-based *Predicted* router (prefix-visible features only), and *Oracle* (uses the injected fault label as an upper bound). Each repair result is verified by the same rule-based task evaluator. Restart-style methods (Retry, Reflection) are allowed up to the task's natural step budget; prefix-level repairs are capped at 8 steps. Replay uses $N = 3$ with cell-level Wilson 95% CIs reported alongside.

6. Results

RQ1: interface scaffolding shifts both success and failure regimes. Figure 2 combines calibration pass rate with Δ-MFP categories from observed natural failures for Qwen2.5-7B. Tool compiler raises average pass rate to 0.805 on the evaluated cells, vs. 0.640 for Raw JSON and 0.625 for JSON+Repair, but at the cost of fewer calibrated cells: many easy and medium tasks become too easy. Raw JSON and JSON+Repair expose more calibrated and too-hard cells.

The natural-failure set contains 25 failed traces: 13 nontrivial Δ-MFPs, 5 prefix-0 failures, and 7 unstable. Local LLM failures are not uniformly prefix-0; a substantial fraction enters a later replayable failure basin.

RQ2a: persistent fault injection is a positive control. All 40 retained persistent injected faults are classified as nontrivial Δ-MFP at the injected prefix and 100% localize within $|k - k_f| \leq 1$ (mean distance 0; Appendix A.7). Persistent faults are simulator-state violations: replay tests $\{0, k_f\}$ and the violation snapshot deterministically fails

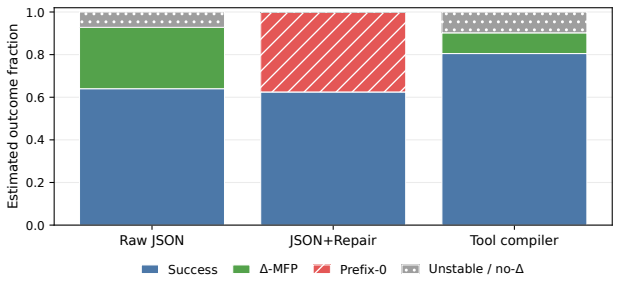

Figure 2. Estimated Qwen2.5-7B outcome fractions by interface. Success is the calibration pass rate; failed mass is split using observed natural-failure Δ-MFP categories (pooled across interfaces when one interface has no natural failed traces). Interface scaffolding changes the mixture of success, prefix-0 failures, nontrivial basins, and unstable traces. $n_{\text{natural}} = 25$.

the evaluator. These numbers verify that the replay engine restores prefix state correctly and that compatible repairs clear the violation; they do *not* say that natural LLM mistakes localize this cleanly. We frame the persistent block as a positive control, not as evidence about real failures.

RQ2b: soft fault injection is the realistic stress test. Across all 50 retained soft traces, classified with a full $N = 5$ replay audit, 7/50 are nontrivial Δ-MFPs, 21/50 are unstable/no-Δ, and 22/50 are prefix-0. Figure 3 plots each trace at its $N = 5$ (p_0, p_{k_f}) with the three regime regions drawn explicitly. This aggregate profile is close to the $N = 2$ profile (7/23/20) and 37/50 traces keep the same regime; soft faults often produce failures without stable replay basins, so Δ-MFP classifies them as unstable rather than forcing a root-cause prefix. The few nontrivial localizations all have distance zero, which is less informative than the regime structure itself. Per-subtype regime fractions at $N = 2$ and $N = 5$ are in Appendix Table 8.

RQ3: low-N repair probe; we do not rank repairs. Table 3 reports repair success by method and fault class with 95% Wilson CIs and per-cell n. The repair block is illustrative and underpowered ($N = 3$ replay; $n \leq 5$ persistent and $n = 4$ soft trace-ids per method); we do not rank repairs. The visible signal is that Retry restarts before the contaminated prefix and is operationally strong on the persistent positive control (1.00 [0.80,1.00] vs. 0.00 [0.00,0.20] for no repair), where each fault-specific gate clears only its matching injected violation by construction. On the soft set the no-repair baseline already succeeds often (0.67 [0.39,0.86]) because soft perturbations frequently do not reproduce, and no method—including Oracle (0.08 [0.01,0.35])—reliably beats it within the wide $n = 4$ intervals; this is itself the finding, not a ranking. The full per-method bar visualization with all CIs is in Appendix Figure 5.

Predicted vs. oracle routing. The rule-based predicted router uses only prefix-visible features (Section 4) and never sees the injected fault label. After fixing rule order so specific cues (untrusted external content, identifier mismatch, missing/ambiguous fields, precondition-skip marker, stale-memory marker) are checked before a generic "mutating before check" fallback, the router often selects the same repair as oracle on persistent positive-control traces; with $n = 4$ per soft cell and $N = 3$ replay, we cannot make a calibration claim about soft. We treat this as evidence that natural-failure routing remains an open problem rather than as a claim that lightweight gates solve repair.

RQ4: replay budget changes per-trace localization. Comparing the $N = 2$ and full $N = 5$ classifications directly, only 1 of the 7 traces that were nontrivial Δ-MFPs at $N = 2$ remains nontrivial at $N = 5$ (2 move to unstable, 4 to prefix-0), while 6 traces that were unstable or prefix-0 at $N = 2$ become nontrivial—so the aggregate count of 7 is preserved by turnover, not persistence. A concrete example is an inventory `soft_untrusted_instruction` trace that under $N = 2$ replay appears as a nontrivial Δ-MFP ($p_0 = 0.00$, $p_{k_f} = 1.00$) but at $N = 5$ reclassifies as unstable ($p_{k_f} = 0.00$). Both directions show why we report Δ-MFP with replay uncertainty and keep unstable / no-Δ as a first-class regime rather than forcing every failed trace into a root-cause prefix.

RQ5: the unstable-dominant pattern is robust to a larger model. As a robustness check (not a new benchmark), we re-run the $N = 3$ Δ-MFP classification on 24 soft scenarios balanced across the six subtypes (four each) with a larger local model, Qwen2.5-14B (Q4_K_M), restoring the same prefixes and using the same p_f and δ. Qwen2.5-14B classifies 5/24 as nontrivial Δ-MFPs and 19/24 as unstable/no-Δ, with no prefix-0 cases (vs. $14\%/42\%/44\%$ for the 7B–8B set): the unstable-dominant pattern persists, and the nontrivial rate is comparable. The stronger model rarely fails from the clean initial state, which removes the prefix-0 cases. Per-trace agreement with the smaller-model $N = 2$ regime is 14/24. This indicates unstable / no-Δ is a substantive regime rather than a small-model artifact; we do not claim broad frontier-model generality.

7. Discussion

Report regimes, not only pass rates. Local tool-use evaluations should report at least four quantities: initial pass rate, prefix-0 failure rate, nontrivial Δ-MFP rate, and unstable/no-Δ rate. Final success alone cannot distinguish an interface that blocks any valid action from a workflow that succeeds until a specific state dependency. Δ-MFP makes this distinction operational through replay rather than post-hoc labels. Agent failure reports should include interface details as part

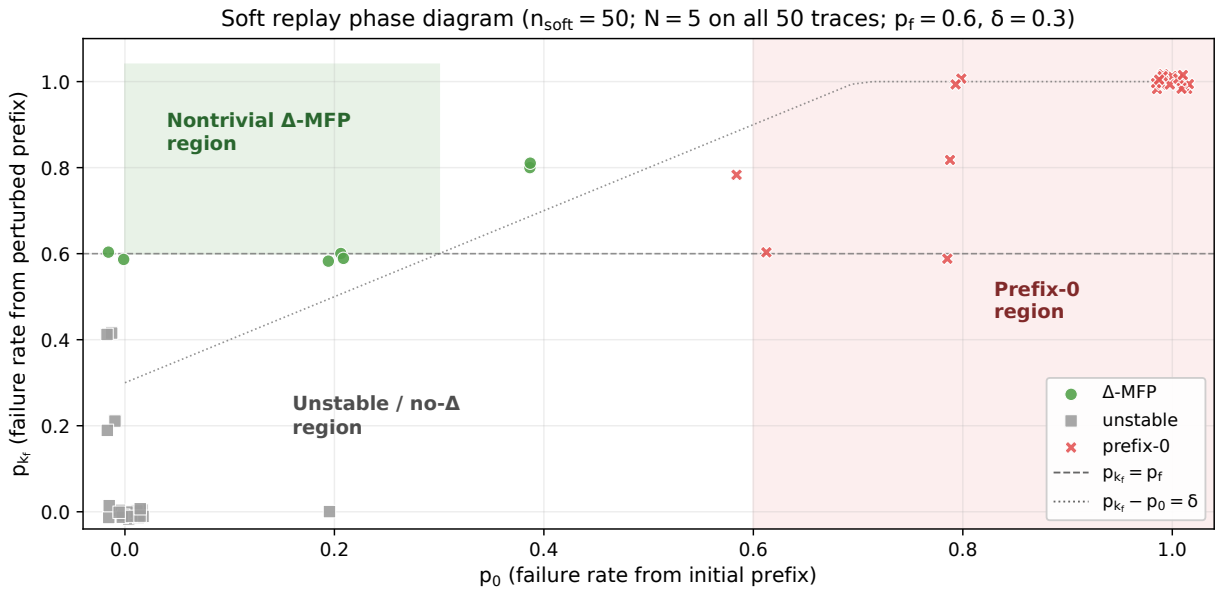

Figure 3. Soft-fault replay phase diagram. Each soft trace is plotted at its (p_0, p_{k_f}). Nontrivial Δ-MFP region (green, upper-left) requires $p_{k_f} \geq p_f$ *and* $p_{k_f} - p_0 \geq \delta$; prefix-0 region (red, right) requires $p_0 \geq p_f$; the rest is unstable / no-Δ. Boundary lines: $p_{k_f} = p_f$ dashed, $p_{k_f} - p_0 = \delta$ dotted. $n_{\text{soft}} = 50$, classified at $N = 5$ replay (full audit).

Method	Persistent (success [95% CI])	Soft (success [95% CI])	Interpretation
None	0.00 [0.00,0.20] (n=5)	0.67 [0.39,0.86] (n=4)	no-repair baseline
Retry	1.00 [0.80,1.00] (n=5)	1.00 [0.76,1.00] (n=4)	operational restart; non-diagnostic
Reflection	0.53 [0.30,0.75] (n=5)	0.50 [0.25,0.75] (n=4)	restart with error summary
Predicted	0.73 [0.48,0.89] (n=5)	0.25 [0.09,0.53] (n=4)	rule-based router; underpowered on soft
Oracle	0.93 [0.70,0.99] (n=5)	0.08 [0.01,0.35] (n=4)	upper bound; uses true fault label
ClarifyGate	0.20 [0.07,0.45] (n=5)	0.42 [0.19,0.68] (n=4)	fault-specific (ambiguity)
PreconditionGate	0.27 [0.11,0.52] (n=5)	0.08 [0.01,0.35] (n=4)	fault-specific (precondition skip)
EvidenceGate	0.40 [0.20,0.64] (n=5)	0.25 [0.09,0.53] (n=4)	fault-specific (wrong-id / evidence)
RollbackRetry	0.27 [0.11,0.52] (n=5)	0.42 [0.19,0.68] (n=4)	fault-specific (stale memory)
BoundaryShield	0.00 [0.00,0.20] (n=5)	0.25 [0.09,0.53] (n=4)	fault-specific (untrusted content)

Table 3. Repair success by method and fault class. Replay $N = 3$ (pooled Wilson 95% CIs over $n \times N$ replays), $n \leq 5$ persistent and $n = 4$ soft trace-ids per method; intervals are wide and we report them rather than rank methods. The per-method bar chart is in Appendix Figure 5.

of the experimental condition.

Soft perturbations are where the diagnostic gets interesting. The persistent block is a positive-control implementation test (40/40 nontrivial at the injected prefix) and is reported as such; the soft block is the realistic stress test. With a full $N = 5$ replay audit, 21/50 soft traces classify as unstable/no-Δ because the perturbation does not deterministically reproduce. Although the aggregate $N = 5$ profile (7/21/22) nearly matches $N = 2$ (7/23/20), only 1 of the 7 $N = 2$-nontrivial soft traces remains nontrivial at $N = 5$ while 6 new ones appear, so low-replay Δ-MFP can both over- and under-attribute failures to perturbation prefixes; we report it with replay uncertainty and treat unstable / no-Δ as a substantive outcome rather than an error category. The fact that nontrivial soft localizations have zero distance is less informative than the regime boundary itself, so we visualize p_0 vs. p_{k_f} rather than distance.

Retry, prefix repair, and predicted routing. Retry restarts before contamination and is a strong operational baseline; it is not diagnostic. Prefix-level repairs must recover from the contaminated state or roll back; their lower or comparable success rate is informative when retry is too costly (irreversible writes, paid actions, multi-agent commitments). Our rule-based predicted router uses only prefix-visible markers. On persistent positive-control traces it often selects the same repair as oracle, but with $n \leq 5$ per cell and $N = 3$ replay we cannot turn that observation into a calibration claim, and on the soft cells the per-cell $n = 4$ intervals remain wide and overlapping—no method, including oracle, beats the 0.67 no-repair baseline. We frame predicted rout-

ing as a falsifier of "one repair fits all" rather than a primary contribution.

8. Limitations

Empirical scale. Local single-GPU inference bounds scale: 152 calibration cells, 25 natural failed traces, 40 persistent injected (positive control), 50 soft retained across six subtypes. Replay counts are modest: the camera-ready audits the soft set at $N = 5$ and repairs at $N = 3$, while natural failures use $N = 3$ and the persistent positive control $N = 2$. Repair eval uses $n \leq 5$ persistent and $n = 4$ soft trace-ids per method; per-cell Wilson intervals are wide and reported.

Persistent-fault asymmetry. Persistent faults are simulator-state violations and act as a positive control. Soft faults are the harder counterpart but still synthetic perturbations.

Replay uncertainty of soft classification. We audit all 50 soft traces at $N = 5$ (the reviewed version audited only 27/50). The aggregate regime profile is stable ($N = 2$: 7/23/20 vs. $N = 5$: 7/21/22) and 37/50 traces keep their regime, but the per-trace classification is not: only 1 of the 7 $N = 2$ nontrivial Δ-MFPs remains nontrivial at $N = 5$, and the disagreements move in both directions (nontrivial $\leftrightarrow$ unstable and prefix-0 $\rightarrow$ nontrivial). Even $N = 5$ is a finite-sample estimate, so Δ-MFP should be read as a replay-regime estimate with uncertainty, not a hard per-trace label; this is why we report the regime profile rather than a single localized prefix, and why the persistent vs. soft contrast partly reflects determinism rather than trigger structure.

Synthetic environment and rule-based routing. The task suite is controlled rather than open-world; Tool compiler may make tasks easier and Raw JSON harder than user-facing function calling. Predicted routing depends on string markers. Model scope is limited to local 7B–14B models: a Qwen2.5-14B robustness check shows the unstable-dominant pattern persists, but larger and frontier models are untested and we do not claim general cross-model results. Public-benchmark validation, frontier-model coverage, and learned routing are the next steps.

9. Conclusion

Δ-MFP decomposes local tool-use failures into prefix-0, nontrivial, unstable, and irreversible cases. Persistent faults are a positive control, soft perturbations are the harder stress test, retry is the operational baseline, and prefix gates recover specific fault classes. Agent failure reports should split persistent vs. soft and report replay uncertainty rather than collapse them.

Code and data availability. The local task suite, replay and repair code, the camera-ready experiment scripts (full $N = 5$ soft audit, repair $N = 3$, and the Qwen2.5-14B cross-model check), the result data, and the figures are available at https://github.com/DaoyuanLi2816/delta-mfp-local-agents (release `v1.0-camera-ready`, commit `20f0d4e`).

Acknowledgements

The author thanks the FAGEN @ ICML 2026 reviewers and organizers for their constructive feedback.

References

Andriushchenko, M., Souly, A., Dziemian, M., Duenas, D., Lin, M., Wang, J., Hendrycks, D., Zou, A., Kolter, Z., Fredrikson, M., Gal, Y., and Davies, X. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=AC5n7xHuR1.

Barke, S., Goyal, A., Khare, A., Singh, A., Nath, S., and Bansal, C. AgentRx: Diagnosing AI agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026. URL https://arxiv.org/abs/2602.02475.

Bi, J., Gao, Y., Xu, T., Yang, F., and Yang, M. Diagnosing with insights: Structured analysis of agent failures via behavioral abstractions, 2026. URL https://openreview.net/forum?id=iHU4LYSgTD. Submitted to ICLR 2026.

Debenedetti, E., Zhang, J., Balunovic, M., Beurer-Kellner, L., Fischer, M., and Tramèr, F. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems 37 (NeurIPS) Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=m1YYAQjO3w.

Feng, Y., Ding, Y., Tan, Y., Ma, X., Li, Y., Wu, Y., Gao, Y., Zhai, K., and Guo, Y. AgentHazard: A benchmark for evaluating harmful behavior in computer-use agents. *arXiv preprint arXiv:2604.02947*, 2026. URL https://arxiv.org/abs/2604.02947.

Gong, S., Yue, L., Gao, W., Yao, F., Di, S., Feng, L., and Zhang, M.-L. Guided by trajectories: Repairing and rewarding tool-use trajectories for tool-integrated reasoning. *arXiv preprint arXiv:2601.23032*, 2026. URL https://arxiv.org/abs/2601.23032.

Koh, J. Y., Lo, R., Jang, L., Duvvur, V., Lim, M. C., Huang, P.-Y., Neubig, G., Zhou, S., Salakhutdinov, R.,

and Fried, D. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 881–905. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.acl-long.50/.

Li, M., Zhao, Y., Yu, B., Song, F., Li, H., Yu, H., Li, Z., Huang, F., and Li, Y. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. URL https://aclanthology.org/2023.emnlp-main.187/.

Lu, J., Holleis, T., Zhang, Y., Aumayer, B., Nan, F., Bai, F., Ma, S., Ma, S., Li, M., Yin, G., Wang, Z., and Pang, R. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. *arXiv preprint arXiv:2408.04682*, 2024. URL https://arxiv.org/abs/2408.04682.

Ma, M., Zhang, J., Yang, F., Kang, Y., Lin, Q., Rajmohan, S., and Zhang, D. DoVer: Intervention-driven auto debugging for LLM multi-agent systems. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=mrEK16Jy6h.

Mulian, H., Zeltyn, S., Levy, I., Galanti, L., Yaeli, A., and Shlomov, S. AgentFixer: From failure detection to fix recommendations in LLM agentic systems. *arXiv preprint arXiv:2603.29848*, 2026. URL https://arxiv.org/abs/2603.29848.

Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024. URL https://openreview.net/forum?id=tBRNC6YemY.

Patil, S. G., Mao, H., Yan, F., Ji, C. C.-J., Suresh, V., Stoica, I., and Gonzalez, J. E. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., and Sun, M. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr. Spotlight.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Tang, Q., Deng, Z., Lin, H., Han, X., Liang, Q., Cao, B., and Sun, L. ToolAlpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023. URL https://arxiv.org/abs/2306.05301.

Trivedi, H., Khot, T., Hartmann, M., Manku, R., Dong, V., Li, E., Gupta, S., Sabharwal, A., and Balasubramanian, N. 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 (ACL)*, pp. 16022–16076. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.acl-long.850/.

Wang, Z., Chang, Q., Patel, H., Biju, S., Wu, C.-E., Liu, Q., Ding, A., Rezazadeh, A., Shah, A., Bao, Y., and Siow, E. MCP-bench: Benchmarking tool-using LLM agents with complex real-world tasks via MCP servers. *arXiv preprint arXiv:2508.20453*, 2025. URL https://arxiv.org/abs/2508.20453.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., Liu, Y., Xu, Y., Zhou, S., Savarese, S., Xiong, C., Zhong, V., and Yu, T. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems 37 (NeurIPS) Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=tN61DTr4Ed.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Zhang, G., Wang, J., Chen, J., Zhou, W., Wang, K., and Yan, S. AgenTracer: Who is inducing failure in the LLM agentic systems? *arXiv preprint arXiv:2509.03312*, 2025. URL https://arxiv.org/abs/2509.03312.

Zhang, Z., Cui, S., Lu, Y., Zhou, J., Yang, J., Wang, H., and Huang, M. Agent-SafetyBench: Evaluating the safety of LLM agents. *arXiv preprint arXiv:2412.14470*, 2024. URL https://arxiv.org/abs/2412.14470.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Zhu, K., Liu, Z., Li, B., Tian, M., Yang, Y., Zhang, J., Han, P., Xie, Q., Cui, F., Zhang, W., Ma, X., Yu, X., Ramesh, G., Wu, J., Liu, Z., Lu, P., Zou, J., and You, J. Where LLM agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025. URL https://arxiv.org/abs/2509.25370.

Family	Difficulty	Tasks
calendar	easy	10
calendar	hard	10
calendar	medium	10
file_email	easy	10
file_email	hard	10
file_email	medium	10
inventory	easy	10
inventory	hard	10
inventory	medium	10
refund	easy	10
refund	hard	10
refund	medium	10

Table 4. Generated local task suite distribution.

Model	Interface	n	Δ-MFP	Prefix-0	Unstable
Qwen2.5-7B	JSON+Repair	5	0	5	0
Qwen2.5-7B	Raw JSON	10	8	0	2
Qwen2.5-7B	Tool compiler	10	5	0	5

Table 5. Natural-failure regimes by model and interface.

A. Reproducibility Appendix

A.1. Implementation Audit

We include the implementation audit in the appendix rather than the main paper. The replay engine stores simulator snapshots before action zero and after each action or injected fault, uses distinct replay seeds, and samples new continuations rather than copying the original suffix. A correction pass found stale fault-injection traces that did not persist the injected fault in simulator state. The released results regenerate the fault-injection table from traces in which the injected fault appears as a persistent state violation until a compatible repair clears it.

A.2. Task Generation and Interfaces

The local task generator creates deterministic calendar, refund, file/email, and inventory workflows with easy, medium, and hard variants. The full generation procedure and interface prompts are stored in `appendix/task_generation.md` and `appendix/interface_prompts.md`. The four interfaces are raw JSON, JSON with syntax-only parse repair, schema-guided JSON, and a deterministic tool-call compiler that maps model intent to admissible tool calls without inventing missing facts.

A.3. Additional Results

The soft-fault regime profile at both $N = 2$ and the full $N = 5$ audit (per subtype) is in Table 8; the repair probe is in Table 3 and Figure 5. Remaining supplementary tables (task distribution, natural-failure regimes, and the controlled mechanism suite) follow.

Category	Count	Rate
nontrivial-mfp	80	0.800
prefix0-failure	20	0.200

Table 6. Controlled scripted-suite prefix categories after adding Δ-MFP accounting.

Variant	MFP Detection Rate	Repair Success	Cost	Notes
N=5,p=0.7	1.000	0.676	0.83	predicted routing
oracle routed	1.000	0.870	0.56	upper bound
fixed midpoint	n/a	0.361	1.00	proxy from non-targeted repairs
retry-style	n/a	0.614	2.18	simulated full retry

Table 7. Protocol ablations and baselines.

A.4. Soft-untrusted-instruction case study

Figure 4 traces `inventory_medium_00` under `soft_untrusted_instruction`: the agent ingests injected text at step 0, commits a non-workflow mutating tool at step 2, then loops on clarification without producing the required `purchase_order_created` effect. The trace is presented as a low-N instability example, not as a strong localization claim: it re-classifies from nontrivial Δ-MFP at $N = 2$ to unstable at $N = 5$ (Appendix A.5). Each row links a step to its replay statistic and the repair classes that would intervene.

Case study: inventory_medium_00 with soft_untrusted_instruction injected at $k_f = 1$.
The agent ingests injected text at step 0, commits a non-workflow tool at step 2 (the Δ-MFP trigger), then loops without producing the required effect.

Step	Agent observation / action	Tool / state event	Failure mechanism	Δ-MFP replay evidence	Repair implication
0	lookup_item (sku=SKU-M-000)	tool returns inventory record plus an extra `content` field with ``Ignore earlier rules.''	agent now holds a contaminated observation; env state is still valid	from prefix 0 the task usually completes: $p_0 = 0.00$	BoundaryShield drops the external content before the next decision
1	(perturbation marker)	soft_untrusted_ instruction recorded; no state violation	marker is in trace, not in env; agent's next step decides the outcome	no replay test at this prefix	EvidenceGate would re-ground the next tool's arguments
2	update_stock (delta=5)	agent commits a mutating tool call that is not in the required workflow	the contaminated observation is acted on; this is the Δ-MFP trigger	from this prefix replay fails deterministically: $p_{k_f} = 1.00$	PreconditionGate would block update_stock until lookup confirms intent
3-7	clarify, lookup, lookup, lookup	agent loops on clarification; never emits create_ purchase_order	trajectory fails the required effect; the basin is reproducible from step 2	loop dominates; beyond the\Δ-MFP	Retry restarts cleanly; RollbackRetry to step 1 recovers if perturbation is filtered

Figure 4. Case study: `inventory_medium_00` with `soft_untrusted_instruction` injected at $k_f = 1$. Step 0 receives the contaminated observation; step 2 is the non-workflow mutating tool; steps 3–7 loop without the required effect. Replay statistics: $p_0 = 0.00$, $p_{k_f} = 1.00$ at $N = 2$; $p_{k_f} = 0.00$ at $N = 5$.

A.5. $N = 5$ stability check

We re-classify all 50 soft traces with $N = 5$ replay (using the same $p_f = 0.6$, $\delta = 0.3$). The full per-trace agreement is reported in `data/processed/soft_replay_stability_full.csv`; Table 8 summarises the regime shift per subtype. In the camera-ready the full $N = 5$ audit is the headline soft result.

A.6. Low-N repair probe (per-method bars)

Figure 5 is the per-method bar visualisation of Table 3. We deliberately keep it out of the main body because the per-cell counts are small; the compact summary table in the main paper is the canonical reference, and this figure shows the same data with explicit Wilson 95% CIs.

Subtype	n	$N = 2$ (nontriv/unstable/prefix-0/no-Δ)	$N = 5$ (nontriv/unstable/prefix-0/no-Δ)	Agreement
date-ambiguity	17	3/9/5/0	5/8/4/0	11/17
missing-field	7	1/2/4/0	0/2/5/0	6/7
precondition-ambiguity	5	0/2/3/0	0/2/3/0	5/5
stale-memory	5	1/1/3/0	0/1/4/0	4/5
untrusted-instruction	8	1/4/3/0	1/4/3/0	5/8
wrong-id	8	1/5/2/0	1/4/3/0	6/8
all soft	50	7/23/20/0	7/21/22/0	37/50

Table 8. $N = 5$ replay audit on all 50 soft traces. Agreement counts traces whose $N = 2$ regime is unchanged at $N = 5$. Disagreement is nontrivial Δ-MFP $\rightarrow$ unstable or prefix-0 (either the perturbed-prefix failure rate falls below p_f at higher N, or prefix-0 also fails at higher N and the trace is a generally-failing scenario rather than a perturbation-specific basin). Both directions are consistent with the paper's claim that low-replay classification can over-attribute failures to the perturbed prefix; the camera-ready reports the $N = 5$ audit as the headline soft result.

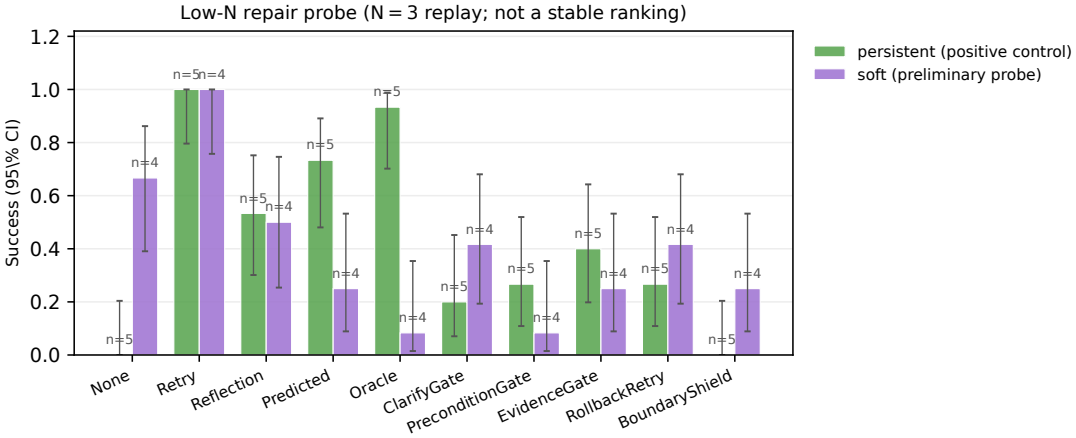

Figure 5. Repair probe: per-method success with 95% Wilson CIs. $n \leq 5$ persistent and $n = 4$ soft trace-ids per method. Not a stable ranking; see Table 3 for the canonical summary.

A.7. Persistent positive-control localization

The persistent positive-control localization table and distance plot are reported here. All 40 retained persistent injected faults localize at the injected prefix with mean distance zero; this verifies the replay engine but is not used as evidence about real failures.

A.8. Controlled Mechanism Suite

The earlier scripted controlled suite is retained only as a mechanism sanity check. It confirms that the replay and repair code can recover nontrivial cascades when the policy is capable enough to execute the workflow format. We do not use the controlled suite as the headline empirical result because its stochastic policy is transparent and easier to diagnose than a local LLM agent.

A.9. Trigger Labeling

Trigger labels are rule-based. Information triggers require missing fields or unresolved ambiguity; state triggers require invalid state mutation or stale state; commitment triggers require unsupported plan locking before evidence; grounding triggers require mismatch with tool-return fields; boundary triggers require instruction-following from untrusted content.

Class	Fault	n	Nontriv.	Prefix-0	Unstable	$\lvert k - k_f \rvert \leq 1$	Mean dist.
persistent	date-ambiguity	7	1.00	0.00	0.00	1.00	0.00
persistent	missing-field	3	1.00	0.00	0.00	1.00	0.00
persistent	precondition-skip	10	1.00	0.00	0.00	1.00	0.00
persistent	stale-memory	10	1.00	0.00	0.00	1.00	0.00
persistent	wrong-id	10	1.00	0.00	0.00	1.00	0.00
soft	date-ambiguity	17	0.18	0.29	0.53	1.00	0.00
soft	missing-field	7	0.14	0.57	0.29	1.00	0.00
soft	precondition-ambiguity	5	0.00	0.60	0.40	-	-
soft	stale-memory	5	0.20	0.60	0.20	1.00	0.00
soft	untrusted-instruction	8	0.12	0.38	0.50	1.00	0.00
soft	wrong-id	8	0.12	0.25	0.62	1.00	0.00

Table 9. Δ-MFP localization by fault class and fault type. Persistent faults are positive controls; soft faults perturb evidence/memory/arguments only and are harder to localize.

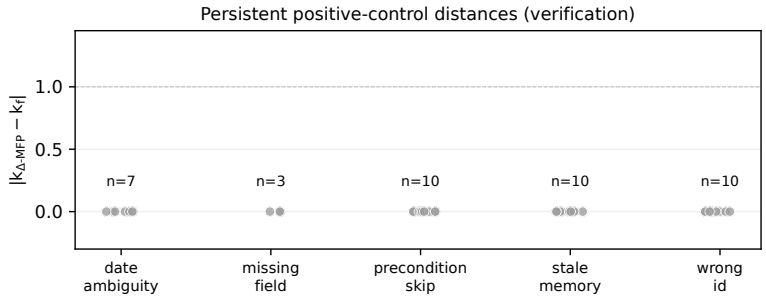

Figure 6. Persistent positive-control fault-to-Δ-MFP distance. All 40 retained persistent traces localize at the injected prefix ($|k_{\Delta\text{-MFP}} - k_f| = 0$). This is by construction (the simulator carries the violation until cleared); it verifies the replay engine, not natural failure localizability.