

Persistent memory for AI coding agents (v0.9.2)

Persistent memory for AI coding agents: a pre-registered SWE-bench Verified benchmark

Author: Saravanan Jaichandaran (independent, world-model-mcp maintainer)

Date: 2026-06-30 (v0.9.2 update; original 2026-06-24)

Version: v0.9.2 (this version adds Appendix A: multi-seed replication, which updates the confidence bounds on the v0.9 headline)

DOI: 10.5281/zenodo.20834509 (Zenodo concept DOI; this is version 2 of the same record)

Code and artifacts: <https://github.com/SaravananJaichandaran/world-model-mcp>

License: Paper CC-BY 4.0; code MIT

Contact: saravananjaichandar@gmail.com

Abstract

AI coding agents lose state at every context compaction. They re-encounter the same failure modes session after session. world-model-mcp is an open-source MCP server that builds a temporal knowledge graph for codebases, capturing facts with provenance metadata and per-evidence-type decay, then re-injecting them after compaction. This paper reports the v0.9 benchmark testing whether that layer measurably reduces repeated coding-agent mistakes on a public corpus.

50 SWE-bench Verified tasks across five repositories (django, sympy, matplotlib, scikit-learn, sphinx) were run as paired baseline-vs-treatment per task. The methodology was committed to DESIGN.md on 2026-06-17, a week before the benchmark ran, so the result is pre-registered and cannot be adjusted post hoc. The treatment arm receives constraints extracted from prior baseline failures via the SWE-bench Pro 7-category failure taxonomy. The agent in both arms is Claude Code 2.1.177 headless.

Across 49 paired SWE-bench Verified instances (1 dropped due to an upstream setup-script issue), baseline pass rate was $33/49 = 67.3$ percent and treatment $38/49 = 77.6$ percent, a delta of +10.2 percentage points. Within-domain delta (django + sympy, constraints from same repo family) was +15.0 pts. Cross-domain delta (matplotlib + scikit-learn + sphinx, constraints loaded only from a different repo family) was +6.9 pts with zero regressions on 18 baseline passes. Six FAIL-to-PASS flips and one regression were observed. Seven explicit limitations are documented verbatim, including single-trial design, within-domain constraint-failure overlap, and judge-model self-reference risk. The result establishes empirical evidence that persistent memory with provenance reduces

coding-agent failure recurrence within-domain, with smaller positive signal cross-domain at no observed cost on the families tested.

Keywords: code generation, AI coding agents, persistent memory, software engineering benchmark, SWE-bench Verified, Model Context Protocol, provenance, learned constraints

1 Introduction

Software engineering agents built on large language models have started to handle real bug-fix and feature tasks in open-source repositories. SWE-bench Verified, a 500-task subset of the SWE-bench corpus curated by OpenAI, has become a common point of reference. Public leaderboards report absolute pass rates. The methodological gap is that almost no benchmarks measure what happens when an agent runs the same kind of task twice. In practice an agent that fails a task today often fails the same kind of task tomorrow, because nothing persists between sessions other than the codebase and the system prompt.

world-model-mcp is a Model Context Protocol (MCP) server that addresses this gap. It hooks into the Claude Code lifecycle events `SessionStart`, `UserPromptSubmit`, `PreToolUse`, `ToolResult`, `PostToolUse`, `PreCompact`, `PostCompact`, and `SessionEnd`, captures facts from agent activity, stores them in a temporal knowledge graph with provenance metadata, and re-injects confidence-weighted facts after compaction. Each fact carries five provenance fields shipped in v0.8.0: `asserted_by` (the tool or agent that wrote it), `confirmer` (a separate tool or human who confirmed it), `confirmation_state` (one of `synthesized`, `corroborated`, `settled`, `canonical`, `superseded`), `evidence_type` (one of `source_code`, `test`, `session`, `user_correction`, `bug_fix`), and `last_decay_at`. A decay function ages each fact by an evidence-type-specific half-life: `test` 180 days, `bug_fix` 365 days, `user_correction` 730 days, `source_code` 365 days, `session` 14 days.

The hypothesis is straightforward. If the agent in session N can read confidence-weighted facts written by the agent in session N-1, the rate at which the same failure mode recurs should drop. The harder question is whether constraints extracted from one repo family transfer to a different repo family, or whether each domain needs its own constraint set.

v0.9 of world-model-mcp ships the empirical test of both questions. This paper reports the methodology, the per-task results, the mechanistic analysis of the two cross-domain flips, and seven explicit limitations.

2 Methodology

Pre-registration. The methodology was committed in `benchmarks/repeat-mistake/DESIGN.md` on 2026-06-17, seven days before the first benchmark run. The committed document specified the hypothesis, the interpretation thresholds (0 to 5 percent delta as null result, 5 to 15 percent as modest signal, 15 to 30 percent as

meaningful signal, 30 percent or more as suspicious), the judge prompts verbatim, the SWE-bench Pro 7-category failure taxonomy, and the sample selection criteria. The file has not been edited since.

Task selection. 50 tasks were drawn from SWE-bench Verified across five repositories with 10 tasks each: django, sympy, matplotlib, scikit-learn, and sphinx. The selection criteria favored repositories with at least 10 verified tasks, documented long-codebase scenarios where memory layers should pay theoretical dividends, and mixed task types so the failure-mode distribution would vary. The final selection is at `benchmarks/repeat-mistake/subset_50.json`.

Two subsets. Subset 1 (n=20) covers django and sympy. Subset 2 (n=30) covers matplotlib, scikit-learn, and sphinx. The intent is to test within-domain and cross-domain transfer separately on the same shared agent and harness.

Two arms per task. Each task is attempted twice, once in the baseline arm and once in the treatment arm. Both arms use the same agent (Claude Code 2.1.177 headless), the same task instance, the same starting commit, and the same test_patch. The only difference between arms is whether constraints extracted from prior baseline failures are loaded into the agent prompt.

Constraint extraction. Each baseline failure is classified using the SWE-bench Pro 7-category failure taxonomy (Wrong Solution, Tool-Use, Syntax Error, Incorrect File, Endless File Reading, Misunderstood Problem Statement, Other) by a Claude judge with a locked prompt. For each Wrong Solution failure, a short directive is extracted via a second locked prompt, intended to address the specific failure mode without leaking the gold answer. The extracted constraints are stored in `constraints.json` for Subset 1 and `constraints_s2.json` for Subset 2.

Cross-domain isolation. For Subset 2, the treatment arm loads only the 4 Subset 1 constraints (django and sympy directives). The 11 Subset 2 constraints, although extracted, are deliberately held out. This isolates the cross-domain transfer effect. The agent sees only out-of-domain directives when attempting matplotlib, scikit-learn, and sphinx tasks.

Scoring. All scoring uses the official SWE-bench harness on SWE-bench Verified, run locally in Docker. Per-task timeout is 1800 seconds. Pass criterion is the standard SWE-bench FAIL_TO_PASS plus PASS_TO_PASS test execution result.

3 Results

Subset 1 (within-domain: django + sympy). 20 paired instances. The 4 constraints used in the treatment arm came from the 5 Subset 1 baseline failures.

Baseline pass rate: $15/20 = 75.0$ percent.

Treatment pass rate: $18/20 = 90.0$ percent.

Delta: +15.0 pts.

Four FAIL-to-PASS flips: django-11400 (RelatedFieldListFilter ordering), django-13212 (validators ValidationError params), django-13344 (recovered from empty patch), sympy-16597 (is_even implies is_finite). One regression: sympy-17630 (no related constraint loaded; failure mode appears unrelated to the four directives).

Of the 5 baseline failures, 4 recovered (80 percent recovery rate) when the constraint matched the failure mode. This is the within-domain upper bound.

Subset 2 (cross-domain: matplotlib + scikit-learn + sphinx). 30 instances, 29 paired. One instance dropped: scikit-learn-25102 fails to build because its setup_repo.sh in the SWE-bench harness uses the pip flag `--no-use-pep517`, which was removed in recent pip versions. The drop applies equally to both arms so the paired comparison stays unbiased on the 29 surviving instances.

The treatment arm loads only the 4 Subset 1 constraints (django and sympy). The 11 Subset 2 constraints are held out.

Baseline pass rate: $18/29 = 62.1$ percent.

Treatment pass rate: $20/29 = 69.0$ percent.

Delta: +6.9 pts.

Two FAIL-to-PASS flips: scikit-learn-14087 (LogisticRegressionCV refit=False indexing) and sphinx-9461 (classmethod + property documentation). Zero regressions across 18 baseline passes.

Combined paired result. 49 paired instances across both subsets.

Baseline: $33/49 = 67.3$ percent.

Treatment: $38/49 = 77.6$ percent.

Delta: +10.2 pts.

Six FAIL-to-PASS flips. One regression. Net +5 paired tasks resolved by the treatment arm.

4 Mechanistic analysis of cross-domain flips

The two cross-domain flips need mechanistic inspection. They are the load-bearing claim that constraints generalize beyond their source repo family.

Flip 1: scikit-learn-14087 (LogisticRegressionCV refit=False IndexError). The baseline patch swaps `self.multi_class` for a local `multi_class` variable but leaves the array indexing wrong for the `multi_class` case. The treatment patch corrects the indexing.

The Subset 1 constraint that may have helped is constraint #3 (sympy classmethod): “update ALL call sites including module-level aliases when changing classmethod semantics, not just the method body.” The mechanistic link is loose. The constraint is about following through on changes across call sites, which is also what the baseline patch

failed to do for the indexing. I do not claim a strong causal mechanism here. This flip could be partially attributable to single-trial variance.

Flip 2: sphinx-9461 (classmethod + property documentation). The baseline patch locates the right files but misses the import-time handling of `@classmethod` `@property` chains. The treatment patch correctly detects the classmethod wrapper and unwraps `__func__` to access the underlying docstring.

The Subset 1 constraint that helped is the same constraint #3 (sympy classmethod). The link is direct. The constraint is about classmethod handling, and sphinx-9461 fails on classmethod wrapper handling specifically. The constraint contains a generalizable insight about classmethod handling that transferred from a sympy context to a sphinx context.

sphinx-9461 is the cleanest mechanistic case in the dataset. scikit-learn-14087 is a weaker case where coincidence cannot be ruled out at this sample size.

5 Limitations

Seven limitations are stated here, not buried in an appendix.

1. Single-trial design. Each task was run once per arm. Some of the observed flips and the one regression may be due to single-trial variance rather than genuine constraint effects. A multi-seed replication would tighten the confidence intervals, but is beyond vo.9 scope.

2. Constraint-failure overlap on Subset 1. The 4 constraints used in the Subset 1 treatment arm were extracted from the 5 Subset 1 baseline failures. The within-domain comparison therefore tests “can the agent fix its own past failures when reminded?” rather than “do constraints generalize?” The within-domain result establishes the upper bound. The cross-domain Subset 2 result is the methodologically clean transfer signal.

3. Cross-domain transfer rate is small. Two flips of eleven baseline failures (18 percent) is positive signal but not a sweeping result. The dataset is too small to claim that cross-domain transfer is reliably positive. The result is consistent with a small positive effect with wide confidence bounds.

4. The zero cross-domain regression finding is fragile. Zero regressions across 18 baseline passes in Subset 2 is the most surprising single finding in this paper, and the one most likely to fail to replicate on a larger or more diverse dataset. I do not claim that out-of-domain constraints are always free.

5. Failure classification uses a Claude judge. The 7-category taxonomy classification was performed by the same model family as the agent under test. A different judge (human, or a different model family) might produce a different category distribution. The SWE-bench Pro paper used GPT-5 as judge with 87 percent human alignment as the precedent. I use Claude because the methodology must be reproducible by anyone with a Claude subscription.

6. Dropped instance. scikit-learn-25102 was dropped because its SWE-bench harness setup_repo.sh calls `pip install --no-use-pep517 --no-build-isolation -e .`, and `--no-use-pep517` was removed from pip in recent versions. The build dies before scoring can begin. This is an upstream SWE-bench harness issue not addressable from this benchmark. The instance was dropped from both baseline and treatment so the paired comparison stays unbiased.

7. Subset selection bias. The 50 tasks were selected with difficulty-weighting within each repo. The reported pass rates are conditional on this selection and should not be compared directly to leaderboard numbers on the full SWE-bench Verified set.

6 Reproducibility

All artifacts to reproduce these results are committed to the repository.

- Task selection: `benchmarks/repeat-mistake/subset_50.json` (50 tasks)
- SWE-bench Verified snapshot: `benchmarks/repeat-mistake/verified.parquet` (SHA-pinned)
- Baseline patches: `baseline_progress.jsonl` plus `baseline_predictions.json` (Subset 1), `baseline_progress_s2.jsonl` plus `baseline_predictions_s2.json` (Subset 2)
- Baseline scores: `baseline_results.jsonl` (Subset 1), `baseline_results_s2.jsonl` (Subset 2)
- Failure classifications: `baseline_classified.jsonl` (Subset 1), `baseline_classified_s2.jsonl` (Subset 2)
- Constraints: `constraints.json` (Subset 1, used in both treatment arms), `constraints_s2.json` (Subset 2, NOT used in v0.9 treatment to keep the cross-domain test clean)
- Treatment patches and scores: `treatment_progress_s1.jsonl`, `treatment_results_s1.jsonl`, `treatment_progress_s2_crossdomain.jsonl`, `treatment_results_s2_crossdomain.jsonl`
- Locked judge prompts: `failure_classifier.py`, `learning_hook.py`
- Pre-registered methodology: `DESIGN.md` (committed 2026-06-17)
- Full results document: `RESULTS.md`

Replication command sequence is listed in the Reproducibility section of `RESULTS.md`. Total agent cost across both arms was approximately 90 USD on a Claude Code subscription. Total wall-clock for scoring on a single Apple M2 Mac with 8 GB RAM was approximately 40 hours including retries and Docker rebuilds.

7 Related work

SWE-bench was introduced in arxiv 2310.06770 (Jimenez et al.). SWE-bench Verified, a curated 500-task subset, was released by OpenAI in 2024. The SWE-bench Pro paper at arxiv 2509.16941 introduced the 7-category failure taxonomy used in this work. Their classifier used GPT-5 as judge with reported 87 percent human alignment. The OpenAI Feb 2026 retrospective found 59.4 percent of SWE-bench Verified test failures trace to test defects rather than model failures. This noise floor affects both arms equally so the paired delta remains signal.

Open-source memory layers for AI agents include memo (vector plus graph hybrid memory), Letta (formerly MemGPT, tiered memory), Zep (temporal knowledge graph with Graphiti backend), Cognee (memory graph generation), and the @modelcontextprotocol Knowledge Graph Memory MCP server (passive JSONL store). Each occupies a distinct position in the design space. world-model-mcp differs in three respects: lifecycle-hook-native capture via Claude Code, an explicit per-fact provenance schema (asserted_by, confirmer, confirmation_state, per-evidence-type decay), and constraint-learning-from-corrections grounded in the SWE-bench Pro failure taxonomy. v0.9 does not include head-to-head comparisons with other memory layers. That is listed as v0.10 scope in DESIGN.md.

Recent platform-vendor moves include OpenAI's Codex Memories (auto-generated local memory files for Codex CLI, opt-in), Anthropic's memory tool (a client-side file-CRUD abstraction for Claude API, with backend implementation delegated to the developer), and continual-training approaches such as Engram (announced 2026-06-24, training compute spent on user context rather than retrieval). world-model-mcp occupies a different layer: retrieval-time memory with provenance, exposed via MCP rather than via the platform-vendor's native API.

8 Conclusion

Persistent memory with provenance, decay, and learned constraints, exposed via MCP and Claude Code lifecycle hooks, produces a measurable improvement in coding-agent failure recovery on a pre-registered SWE-bench Verified benchmark. The effect is strongest within-domain (+15.0 pts, 80 percent recovery rate on baseline failures matched by a constraint). The effect is smaller but present cross-domain (+6.9 pts, 18 percent transfer rate on baseline failures, zero regressions on 18 baseline passes). The combined paired delta across 49 instances is +10.2 pts.

The result bounds the wedge honestly. Within-domain, persistent constraints help substantially when the constraint matches the failure mode. Cross-domain, the effect is smaller and mediated by generalizable insights inside otherwise domain-specific constraints, with the sphinx-9461 flip as the cleanest mechanistic case. Out-of-domain constraints had zero observed cost on this dataset, but this is the finding most likely to fail to replicate at scale.

Future work targets multi-seed replication, larger task counts per repo, head-to-head against other memory layers (memo, Letta, Zep, pii-a-engram), and an explicit failure-mode-similarity scoring to predict when cross-domain transfer will succeed.

Acknowledgments

Comments on the working-group GitHub threads anthropics/claude-code#47023, anthropics/claude-code#30039, and openai/codex#21753 from Patdolitse (pii-a-engram), ferhimedamine (Dakera AI), kcarriedo (Claudeverse), rpelevin, Necmttn, and safal207 (Liminal) helped shape the v0.8.0 provenance schema that this benchmark validates.

References

- SWE-bench (Jimenez et al.): <https://arxiv.org/abs/2310.06770>
- SWE-bench Verified introduction: OpenAI 2024
- SWE-bench Pro failure taxonomy: <https://arxiv.org/abs/2509.16941>
- memo OSS memory layer: <https://memo.ai>
- Letta (formerly MemGPT): <https://letta.com>
- Zep / Graphiti: <https://github.com/getzep/graphiti>
- @modelcontextprotocol/memory reference server: <https://github.com/modelcontextprotocol/servers/tree/main/src/memory>
- world-model-mcp v0.9.1 release: <https://github.com/SaravananJaichandaran/world-model-mcp/releases/tag/v0.9.1>
- world-model-mcp v0.9.2 release: <https://github.com/SaravananJaichandaran/world-model-mcp/releases/tag/v0.9.2>

Appendix A: Multi-seed replication (v0.9.2 update, 2026-06-30)

The v0.9 limitations section identified single-trial design as the primary methodological risk: “Some of the observed flips and the one regression may be due to single-trial variance rather than genuine constraint effects.” This appendix reports the multi-seed replication carried out to bound that risk. It was added to the v0.9.2 release on 2026-06-30 and is published verbatim per the pre-registration in SEED_PLAN.md.

A.1 Pre-registered subset

A pre-registered 17-instance subset was selected from the 49 paired instances of v0.9 and locked in `SEED_PLAN.md` on 2026-06-25, six days before any additional seed run. The subset contains three categories: 7 load-bearing instances (the 6 FAIL-to-PASS flips and the 1 PASS-to-FAIL regression that drove the v0.9 headline), 5 variance-floor PASS instances (tasks that passed in both arms in v0.9, used to characterize the stability of easy outcomes), and 5 variance-floor FAIL instances (tasks that failed in both arms in v0.9). Subset selection, variance metrics, and interpretation thresholds were locked before seed 2 runs began.

A.2 Methodology (unchanged)

Each instance was re-run at seed 2 in both baseline and treatment arms. The agent (Claude Code 2.1.177 headless), the task instance, the starting commit, and the `test_patch` were all identical to the v0.9 runs. Claude Code CLI does not expose a `--seed` or `--temperature` flag, so “multi-seed” here means observing the model’s intrinsic sampling distribution at default temperature by re-running. The treatment arm loaded the same 4 v0.9 constraints from `constraints.json`. No methodology changes.

A.3 Headline result

Per-arm pass rates on the 17-instance subset:

- v0.9 baseline (seed 1): 6 of 17 = 35.3 percent
- Seed 2 baseline: 13 of 17 = 76.5 percent
- v0.9 treatment (seed 1): 11 of 17 = 64.7 percent
- Seed 2 treatment: 12 of 17 = 70.6 percent

The baseline arm pass rate swung +41 percentage points between seed 1 and seed 2 on the same 17 instances, with no methodology change. The treatment arm swung +6 pts over the same window.

Per-seed paired delta on the subset:

- Seed 1 paired delta: +5 instances (+29 pts within the subset)
- Seed 2 paired delta: -1 instance (-5.9 pts within the subset)
- Mean paired delta across both seeds: +0.24 per instance, bootstrap 95 percent confidence interval [0.00, 0.47]

Load-bearing replication: 0 of 7 load-bearing instances had both their seed-1 baseline outcome AND their seed-1 treatment outcome reproduced at seed 2. Per the thresholds locked in `SEED_PLAN.md`, this is weak replication.

A.4 What drove the replication failure

The treatment arm at seed 2 is relatively stable: 5 of 6 v0.9 “treatment PASS” instances remained PASS at seed 2, and the 1 v0.9 “treatment FAIL” (sympy-17630, the regression) reverted to treatment PASS at seed 2. The treatment patches are not the source of the replication failure.

The replication failure is driven by the baseline arm. All 6 v0.9 “baseline FAIL” instances on the load-bearing subset became baseline PASS at seed 2. The single v0.9 “baseline PASS” (sympy-17630) became baseline FAIL at seed 2. The same agent at the same temperature on the same task gave different outcomes between the two seeds. This is intrinsic agent variance, not constraint effect.

A.5 Honest interpretation

The v0.9 +10.2 pts paired delta on 49 paired instances at single trial was substantially inflated by an unlucky baseline draw at seed 1. The mean paired delta across both seeds on the 17-instance subset is +0.24 per instance with a 95 percent confidence interval of [0.00, 0.47]. The constraint effect is small, possibly nonzero, and not statistically distinguishable from zero at sample size 2.

Three secondary observations from seed 2: - Cross-domain transfer is more fragile than v0.9 claimed. sphinx-9461 (the cleanest mechanistic case in v0.9) regressed to treatment FAIL at seed 2; sklearn-14087 replicated. - The v0.9 regression (sympy-17630) was a single-trial artifact. The task is flaky in both arms, not a stable regression caused by the constraints. - A new treatment-side regression appeared at seed 2: matplotlib-14623, a variance-floor PASS instance in v0.9, passed seed 2 baseline but failed seed 2 treatment. The agent’s patch fixed the FAIL_TO_PASS target test but broke 181 PASS_TO_PASS rendering tests. Constraint loading occasionally introduces broad PASS_TO_PASS regressions on previously stable tasks.

A.6 What this changes

Architectural claims of world-model-mcp (lifecycle-hook-based memory capture, per-fact provenance schema, per-evidence-type decay, PreToolUse defer enforcement) are unchanged by the multi-seed result. The empirical claim about the magnitude of the constraint effect on SWE-bench Verified is what changes: the +10.2 pts paired delta from v0.9 should be read as a single-trial upper bound, not as the steady-state effect size. The replicated effect size on this subset across two seeds is small, possibly zero.

The wedge survives this update. The headline number does not.

A.7 Decision on seed 3

Seed 3 was considered and skipped. The 0 of 7 load-bearing replication pattern is clear, not borderline. Additional seeds at the 17-instance subset would tighten the confidence interval from $[0.00, 0.47]$ to perhaps $[0.05, 0.35]$ but would not flip the verdict. Future replication work (for v1.0 or for a peer-reviewed venue submission) should run all 49 paired instances at 3-5 seeds rather than another 17-instance pass.

A.8 Reproducibility (multi-seed)

Multi-seed artifacts are committed alongside the v0.9 artifacts in `benchmarks/repeat-mistake/`:

- `SEED_PLAN.md` : pre-registered methodology, locked 2026-06-25
- `multi_seed_aggregate.py` : variance-analysis script that produced the headline numbers above
- `baseline_progress_seed2.jsonl`,
`treatment_progress_seed2_treatment.jsonl` : seed-2 agent run records
- `baseline_predictions_seed2.json`,
`treatment_predictions_seed2.json` : seed-2 patch predictions for the SWE-bench harness
- `baseline_results_seed2.jsonl`, `treatment_results_seed2.jsonl` : seed-2 harness scoring results
- `multi_seed_summary_seed2.json` : aggregated variance metrics

Total additional agent cost for seed 2 was approximately 53 USD. Total additional wall-clock for seed-2 scoring on a single Apple M2 Mac with 8 GB RAM was approximately 9 hours including Docker daemon restarts for disk reclamation.