The inquiry loop on SWE-bench Pro

Hypothesis graphs as agent semantic memory, grounded in Peircean methodeutics

June Kim

2026-05-28

Contents

Download PDF — arxiv-shape preprint, rebuilt from this source.

0.1 Abstract

- Headline (the pending claim, finalized when the Pro run terminates). This is a draft; the Pro run is in flight. The claim the paper will assert at v1 finalize: *no method documented in our comparative literature search (§11) has proved, with equivalent receipts, a higher SWE-bench official-harness resolve rate at a lower audited per-instance dollar cost, reproducibly on the two most popular SWE benchmarks (Verified and Pro) under one frozen harness, than the fully automatable skill set this paper presents.* Of this claim, the Verified component is proved now (426/438 eligible, public, Zenodo-DOI'd); the Pro component is pre-registered and in-flight. The headline becomes unconditional only when the Pro run terminates and §3.10 provenance is published; until then it is the explicit target the run is testing. The load-bearing properties are *proved* (we publish per-instance trajectories, captured diffs, gate traces, and cost ledger), and *reproducibility on both benches* under a single frozen artifact. "Resolve rate" means eligible WIN / eligible terminal under the official SWE-bench grader. "Cost" means audited per-instance dollars from the published API-mode ledger (subscription mode is a parallel access path for replication, not the comparative cost unit; see §4.5 for the normalization). The receipts grade four different failure modes, not one:
  - Resolve rate, controlled tier. 426 / 438 eligible on SWE-bench Verified (frozen, public, Zenodo-DOI'd, per-instance trajectories and captured diffs committed).
  - Resolve rate, contamination-resistant tier. In-flight run on SWE-bench Pro under preregistration. Current state (2026-05-28): 196 WIN / 8 LOSS = 96.1% on N = 204 terminal grades across 402 of 728 unique eligible instances; 198 INCOMPLETE (re-run queue draining); 326 instances untouched (heavy tail

pending). Per-instance trajectories and captured diffs accumulate in `runs/scored/artifacts/`. Final headline pending run termination and full §3.10 provenance publication.
  - Ecological grading. 80 merged PRs across 46 repositories on the open OSS deployment surface, 53% maintainer-grading merge rate, GraphQL-verifiable.
  - Cost. ~$2 audited per instance (Sonnet generator + GPT-5.5 challenger, blended across light and heavy repos; per-instance ledger published). ~$1k for a full bench run, well inside an individual researcher's discretionary spend. Spot-check sampling (verifying ten of the published WINs end-to-end at the receipt level) costs ~$20 in API credits; *student-sampling* range, no grant or affiliation required. Frontier-comparable reference points: VentureBeat reports GPT-5.5 at ~$5.80/trial median on Pro-class tasks; SWE-rebench reports Cursor Composer 2.5 at $0.23/problem but at lower documented resolve rates. Replication accessible on a single consumer subscription as an alternate channel.

  What "proved" excludes. A submission that claims a higher rate without publishing per-instance trajectories, captured diffs, gate traces, and a cost ledger has not proven it at the receipt level this paper provides. A reader who finds a documented method that has *proven* a higher resolve rate at a lower audited cost can refute the headline by citing it; the headline survives until then.
- What this paper presents. A working agent harness for industrial code, assembled from prior lineages into a typed inquiry loop over a hypothesis-graph semantic memory. The closest contemporary precedent is Voyager (Wang et al. 2023), which closes an observe?hypothesize?test?commit loop in an embodied setting; this work adopts the same loop shape, types its stages explicitly with Peirce's three modes (1878, 1903), gives it a persistent hypothesis-graph smem to read and write, and runs it on SWE-bench Verified and Pro under preregistration. Additional lineages absorbed: bi-abductive program analysis (Calcagno et al. 2009), Vovk-Wang-style sequential evidence accumulation (Vovk & Wang 2021), and invariant-feature splitting under environment variation (Arjovsky et al. 2019). The pieces are not novel; the runnable assembly is.
- Mechanism. Each instance is a problem of inquiry. Recon (abduction) writes hypothesis nodes. Craft (deduction) traces consequence-edges into an intervention. Audit (induction) reads back evidence, kills or witnesses leaves, classifies the trajectory. A deterministic gate consumes the trajectory shape and routes the next move. No model arbitrates its own termination.
- Adversarial filtering at the hypothesis stage, not the patch stage: blind-blind pushout runs two frontier models on the same evidence pack with no cross-visibility; the merge surfaces disagreements as the next investigation edge.
- Validation has three complementary surfaces.
  - Bench. Two SWE-bench tiers under one frozen harness: Verified (saturated, contaminated, ports cleanly as a baseline) and Pro (contamination-resistant, the primary validation surface). Preregistered, frozen by git tag, full per-instance provenance.
  - Wild. The same methodology runs continuously against open-source GitHub repos since the pipeline epoch (2026-05-09), producing PRs that adversarial maintainers grade by merging or rejecting. As of this writing: 53% merge rate across 46 distinct repositories, 80 PRs merged, 67% positive reception on 65 issues filed. Out-of-distribution maintainer-graded evidence, no curation, no test-set.
  - Corpus. A public corpus of ~380 hypothesis graphs committed under `sweep/repo-hypotheses/`, one per investigated issue. Each graph contains abductive hypotheses, supporting evidence, devil's-advocate critiques, and an engage/skip verdict. The corpus is the deployment trace: every PR in the OSS surface above was preceded by a hypothesis-graph triage decision visible in this directory.
- The three surfaces are reported because they grade different failure modes: controlled comparability (bench), maintainer contact (wild), trace inspection (corpus).
- No model training, fine-tuning, or RL. Frontier models are used as-is via API. What the assembly contains is an *organization of inference* (the typed loop, the hypothesis-graph smem, the deterministic gate), not a weights change.
- Cost envelope. The full run on both benches is runnable on a frontier-vendor subscription plan (~$200/month, multi-week pace) or on roughly $1k of API spend (faster, capacity-bounded). Replication requires no training compute, no curated dataset, and no GPU resources. The full per-instance cost ledger is published alongside the trajectories.
- Lineage. The argument draws on dispersed primary sources: Peirce 1878 and 1903 on the irreducible three modes; Wald 1947, Vovk and Wang 2021, and Ramdas 2023 on sequential evidence accumulation; Calcagno et al. 2009 on bi-abductive program analysis; Wang et al. 2023 (Voyager) and Arjovsky et al. 2019 (IRM) as

cross-field witnesses to the loop pattern. The synthesis happens at the harness level; the citations are not novel and not ours.

0.2 1. Introduction

Agents acting on engineered systems have nowhere to put what they currently believe is going on. Prose context is lossy; skill libraries store verified code, not falsifiable claims; vector retrieval indexes documents, not hypotheses.

Graph-memory work for agents (AriGraph, Anokhin et al. 2024; CausaLab, Yang et al. 2026; CoALA, Sumers et al. 2023) builds adjacent infrastructure without imposing typed reasoning-mode contracts on read and write. The closest SE-targeted precedent is CMM (Khalid & Arora 2026, *Cognitive Memory Manager*, published one day before this draft), which captures coding-agent sessions as a typed reasoning DAG but types nodes by trajectory role rather than reasoning mode, and gates on human approval plus confidence decay rather than deterministic finite-state operations. The belief substrate remains undertyped.

This paper presents a persistent typed hypothesis graph for LLM-agent semantic memory. Peircean reasoning modes are first-class node types. Falsification criteria are executable edge predicates. Pipeline stages have mode-specific write contracts. Routing and gating are performed by deterministic graph operations rather than model calls.

The specific composition is tri-disciplinary by construction: it synthesizes Peircean epistemology (1878, 1903), the cognitive-architecture memory tradition (Soar; Laird 1987; ACT-R; chunking-style mechanical operations on structured memory), and LLM-agent read/write semantics. Graph memory for agents is old; Soar has had it since 1987. We borrow Soar's memory typology (semantic memory / `smem`, procedural memory / `pmem`, episodic memory / `epmem`) only as vocabulary. The hypothesis graph occupies the `smem` slot, the pipeline-stage skills occupy the `pmem` slot, the per-run trajectories occupy the `epmem` slot. The fuller mapping is in §8.2b. Our search (§11) did not find a prior `smem` instance that combines LLM-shaped prose read/write, Peirce-typed falsifiable nodes, Vovk-Wang e-value kill-conditioned edges, one-mode-per-stage write contracts, and model-free deterministic gate operations.

The inquiry loop is the typed read/write contract over that smem. Each stage is constrained to one Peircean mode (1878, 1903): recon writes only abductive nodes, craft reads abductive nodes and emits deductive consequence-edges, and audit reads predictions and writes inductive verdicts plus a trajectory classification. Stages cannot freelance into each other's mode. The Peircean vocabulary names the contract surface; the pipeline enforces the contract mechanically, not the philosophy.

The headline claim is narrow and falsifiable, and conditional on the in-flight Pro run for full unconditional form. No method documented in our comparative literature search (§11) has *proved*, with equivalent receipts, a higher SWE-bench official-harness resolve rate at a lower audited per-instance dollar cost, reproducibly on the two most popular SWE benchmarks (Verified and Pro) under one frozen harness, than the skill set this paper presents.

The four receipts grade different failure modes:

- Verified resolve rate (426 / 438 eligible, Zenodo-DOI'd, trajectories and diffs committed): controlled-bench accuracy.
- In-flight Pro under preregistration (current state 2026-05-28: 196/8 = 96.1% on 204 terminal grades, 402/728 touched, 326 untouched): contamination-resistant accuracy.
- OSS PR merge rate (80 merged across 46 repos, 53%, GraphQL-verifiable): ecological maintainer grading.
- Sub-$1k audited per-instance API cost (ledger published): accessibility.

The load-bearing properties are *proved* (we publish the trail that makes the numbers auditable) and *reproducibility on both benches* under one artifact. Refutation is a citation showing a stronger combined receipt.

Three independently developed recent papers reach for typed-reasoning primitives for LLM agents. IDEA (He et al. 2025, ACL Findings) applies Peirce-cited rule learning outside SE. ADI (Gilda & Gilda 2026, April 17) targets algebraic invariants rather than SE. CMM (Khalid & Arora 2026, May 26) is a typed coding-agent reasoning DAG with cross-session skill graduation; this is the closest SE-adjacent work. Older lineage: Voyager (Wang et al. 2023) closes an empirical observe?hypothesize?test?commit loop in Minecraft without typing the stages; AriGraph

(Anokhin et al. 2024) and CausaLab (Yang et al. 2026) build graph memory for agents in non-SE domains; CoALA (Sumers et al. 2023) is the broader cognitive-architecture framework. Our search (documented in §11) did not surface a prior treatment combining SE-agent harness, Peircean-typed nodes, kill-conditioned edges, one-mode-per-stage write contracts, deterministic gating, and full per-instance provenance. The assembly is what this paper exhibits.

Adversarial filtering operates at the hypothesis stage. Blind-blind pushout runs two frontier models from different families with no cross-visibility, then merges on disagreement while the worktree is still untouched. Agreement is uninformative; disagreement supplies the next investigation edge. Termination is mechanical: a deterministic gate consumes the audit-classified evidence trajectory (Wald 1947, Vovk & Wang 2021) and routes re-entry, completion, or budget exhaustion. The gate is finite-state over trajectory shape, attempt count, budget, and frontier-closure status. No model sits in the gate.

SWE-bench is the legibility surface. Two tiers run under one frozen harness: Verified (saturated, contaminated, ports cleanly as a baseline) and Pro (contamination-resistant, the primary validation surface). Same frozen loop, two contamination regimes, two independent provenance trails. The bench supplies legibility; the methodeutics is the contribution.

The contribution is positional, not foundational. No primitive is introduced. The pieces (Peirce's three modes, bi-abduction, Vovk-Wang e-values, Pearl's DAG, Ramsey's credence, Voyager, IRM, Soar-lineage structured agent memory) are not ours. The runnable assembly is what the paper exhibits; the harness executes on industrial code under contamination discipline, or it doesn't. The comparison with the closest SE precedent (CMM) and the parallel Peirce-typing work (IDEA, ADI) is laid out in §8. CMM is a compsci-grounded synthesis that reaches an adjacent intersection without crossing into either the Peircean or Soar-cogsci lineages.

The hypothesis graph itself decomposes into three independent primitives from three lineages. The structural skeleton is Pearl 1988, 2000: the DAG is Pearl's representation primitive for structured belief. We borrow the structure (directed, acyclic, typed nodes with typed edges), not the semantics (probabilistic conditional independence, do-calculus). Nodes here are typed hypotheses; edges encode dependence and the kill conditions that fire on evidence.

The node semantics is credence under stakes (Ramsey 1926, *Truth and Probability*; James 1907; Dewey 1929). Ramsey's operational credence: a belief is what you'd bet on, its strength is the odds, and the threshold for treating it as actionable is set by the stakes. The pragmatist lineage (Peirce, James, Dewey) makes truth inseparable from action. Each hypothesis-graph node therefore carries a *belief* at a credence level, stakes-indexed and continuous, not a fact. Confident confabulation, high apparent confidence over a node that has not earned it, is the named failure mode that the Peircean stage-typing and the kill-condition discipline are jointly designed to prevent. The reasoning-mode label types *what kind* of belief the node carries; the credence types *how much*.

The update semantics is qualitative trajectory-shape classification, drawn from the sequential-evidence and dynamical-systems lineages (Ville 1939 ? Wald 1947 ? Vovk & Wang 2021 ? Shafer 2021 ? Ramdas 2023 for the temporal-evidence side; Milnor 1985 and Strogatz 2014 for the shape taxonomy). Across audit re-entries, the evidence pattern for a node is labeled with one of four shapes (convergent, divergent, oscillatory, chaotic; §3.4), and the shape label fires the kill or witness decision. A deployed numerical e-value accumulator with a fixed two-sided threshold would require IID Bernoulli test outcomes, which SWE-bench audits do not satisfy; v1 takes the qualitative taxonomy and defers the numerical accumulator to a later version where the IID assumption can be earned.

Pearl plus credence-under-stakes plus trajectory-shape classification: three primitives, three lineages, three roles. They enter the design through separate roles (structure, node semantics, update semantics), and the data structure as implemented requires all three. The provenance is independent; the design integrates them.

0.3 2. Theoretical grounding: methodeutics as a contract surface

Peirce's *Illustrations of the Logic of Science* (1878) and *Pragmatism as the Logic of Abduction* (1903) type the operations of inquiry into three modes that are not interchangeable. Abduction generates explanatory hypotheses for surprising observations: *what would, if true, make this no longer surprising?* Deduction derives testable predic-

tions from hypotheses: *if this hypothesis holds, what follows?* Induction tests predictions against evidence: *does the evidence accord with the prediction?*

The modes are typed by what they cannot do. Abduction proposes content but does not test it; induction tests but introduces no new explanatory content; deduction traces consequences but invents nothing. Collapsing them produces familiar failure modes: confirmation bias (induction without abductive alternatives), confabulation (abduction without inductive grounding), free-association (no typed mode at all). Modern LLM agents collapse the three by default; a single forward pass proposes, predicts, evaluates, and rationalizes in undifferentiated prose.

Methodeutics is Peirce's term for the methodology of inquiry, the systematic study of how to conduct the typed-mode loop well. The framework draws on a dispersed primary-source lineage that this paper cites directly rather than through any single secondary work.

Modes of reason and the irreducible three. Peirce 1878, 1903; Bacon 1620 (induction); Popper 1934 (falsification); Meehl 1967 (the "soft science" critique); Feynman 1974 (cargo-cult science).

Pragmatist credence and stakes-indexed belief. Ramsey 1926 (*Truth and Probability*; subjective probability, the Dutch Book argument, belief as betting odds); James 1907 (*Pragmatism*; truth as what works); Dewey 1929 (*The Quest for Certainty*; truth as warranted assertibility). The node-level semantics of the hypothesis graph descends from this lineage.

Bi-abduction, tri-abduction, and compositional inference. Bylander et al. 1991 (abductive complexity); Calcagno et al. 2009 and O'Hearn 2019 (bi-abductive shape analysis; the anti-frame/frame decomposition for sequential composition, scaled industrially in Facebook Infer); Noam Zilberstein, Saliling & Silva 2024 (arXiv:2305.04842, OOPSLA), *Outcome Separation Logic*, which extends the bi-abductive lineage to branching composition under nondeterministic and probabilistic effects (their Theorem 5.1: soundness of tri-abduction for reconciling two branch preconditions). We use this as a branch-composition analogue, not as a completeness result for arbitrary abduction.

Evidence trajectories and anytime-valid inference. de Finetti 1937 (subjective probability); Ville 1939 (martingales); Wald 1947 (sequential testing); Robbins 1967; Chernoff 1959; Kelly 1996 (capital-growth ties to e-values); Shafer 2021; Grünwald 2024; Ramdas 2023.

Four-bin dynamical classification. Milnor 1985; Strogatz 2014.

Directed acyclic graphs as reasoning representation. Pearl 1988 (*Probabilistic Reasoning in Intelligent Systems*; Bayesian networks as DAGs of dependencies); Pearl 2000/2009 (*Causality*; structural causal models, d-separation, do-calculus). The data structure we use (typed nodes with directed edges, no cycles) is Pearl's lineage applied to hypothesis representation rather than to causal-structure inference.

Working instantiations of the loop in different fields, with no shared lineage between them. Calcagno et al. 2009 (Facebook Infer); Wang et al. 2023 (Voyager); Arjovsky et al. 2019 (Invariant Risk Minimization).

The loop shape recurs across fields with zero shared citations between rediscoveries, and the pre-2025 instances do not cite Peirce at all. *Facebook Infer* (Calcagno et al. 2009) does bi-abductive shape analysis at industrial scale. *Invariant Risk Minimization* (Arjovsky et al. 2019) is tri-abductive figure-ground splitting under environment variation. *Voyager* (Wang et al. 2023) closes an observe?hypothesize?test?commit loop in Minecraft with a skill library, untyped at the node level. *IDEA* (He et al. 2025, ACL Findings) brings the first-class Peirce citation for LLM rule-learning agents but does not target SE. *ADI* (Gilda & Gilda, arXiv:2604.15727, 2026-04-17) develops Peircean tripartite reasoning over a symbolic knowledge graph, six weeks before this draft. *CMM* (Khalid & Arora, OpenReview, Agent Skills '26, 2026-05-26) is the closest SE-targeted precedent: a typed reasoning DAG for coding-agent sessions, published one day before this draft, independently developed, with trajectory-role typing rather than Peirce-typing, no kill-condition edges, no one-mode-per-stage contracts, and a human-approval gate.

Six independent arrivals across seventeen years; the last three within roughly a year of each other; CMM and this paper within twenty-four hours. We treat this as suggestive context for why the typed-reasoning move keeps recurring, not as proof. The framework's public exposition predates CMM and ADI; the dated blog-post timeline is in §10. The cell-by-cell comparison with CMM (typing axis, edge semantics, gate, temporal scope) is in §8.2b.

0.4 3. Method

Figure 1. The inquiry-loop harness used on SWE-bench Pro. Stages are typed by Peircean mode (recon = abduction, craft = deduction, audit = induction); the gate is finite-state with no model call; the outer loop re-enters recon with an updated graph rather than retrying the patch.

0.4.1 3.1 The inquiry frame

Each SWE-bench instance is recast as an inquiry on an engineered system: a failure trace, a codebase, a question of root cause, and an intervention that must not regress the rest of the system.

Code is the right substrate for this data structure because it combines three properties that other inquiry domains rarely combine. It is *reproducible* (same input yields same output, modulo controlled nondeterminism), *deterministic* (causal lines from input to behavior are mechanical), and *perturbable* (single-line and single-function diffs are cheap to apply and fully observable). Hypotheses about code can therefore be tested by cheap mechanical perturbations and falsified by deterministic predicates; kill conditions are not approximations, they are executions.

One instance is enough to establish causality in this regime. Statistics is the machinery for inferring causal relationships from noisy populations; in code the per-instance response is mechanically observable, so a single passing test on a captured diff is a complete causal demonstration for the specified executable predicate on that instance. Hidden behaviors not covered by the executable predicate are out of scope; we report only what the predicate checks, which is what the grader checks. Medical hypotheses need populations because individual responses are noisy; code does not. The paper therefore reports counts and denominators rather than confidence intervals, hypothesis tests, or significance levels: the per-predicate per-instance verdict is already exact, and aggregating across instances is bookkeeping, not inference. Portability of the data structure to substrates without these properties is open.

The three Peircean modes become three stages with disjoint contracts and disjoint access patterns over the hypothesis graph. Abduction (recon) proposes candidate causes and writes hypothesis nodes with falsifiable predicates and kill conditions; no edits, no external access. Deduction (craft) traces each surviving hypothesis's consequences into a concrete patch; an adversarial challenger critiques the diff against the spec. Induction (audit) runs the test suite and classifies the result as an evidence trajectory whose shape determines the next move, not as pass/fail.

0.4.2 3.2 Hypothesis graph as recon output

Recon emits a hypothesis graph document, not a patch sketch. The structure is a three-pillar synthesis. The *skeleton* (Pearl 1988, 2000) is a directed acyclic graph: nodes are typed hypotheses, edges carry dependence and kill predicates. The *node semantics* is credence under stakes: each node carries a stakes-indexed credence in its claim, not a fact; the Peircean reasoning-mode label types what kind of belief, the credence types how much. The *update semantics* is qualitative trajectory-shape classification (Wald 1947; Vovk & Wang 2021; Ramdas 2023; Milnor 1985): across audit re-entries the evidence pattern is labeled convergent / divergent / oscillatory / chaotic, and the shape label fires the kill condition. §3.4 details the four shapes and why a deployed numerical e-value accumulator is set aside in v1 (IID violation).

Kill conditions are mechanical predicates over the evidence trajectory, not model preferences. A node dies when its predicate fires. The graph persists across iterations; re-entry adds nodes rather than overwriting. The frontier closes when every leaf is killed or witnessed.

Worked example. Drawn from `kimjune01/sweep/repo-hypotheses/antonmedv__fx__413.md` (one of the ~380 publicly committed graphs from the OSS deployment surface), abridged. The issue: under the `snap`-installed build of `fx`, the yank-to-clipboard keystroke flashes the dialog but the clipboard stays empty. An early recon pass converged on a documentation-only verdict; a later pass re-entered the graph and surfaced a shippable code fix the earlier passes had not generated. The graph (compressed):

```
H0: snap strict confinement blocks fx's clipboard subprocess
  Perturbation surface: main.go:645-660 (yank handler), snap/snapcraft.yaml
  Falsifiable predicate: clipboard.WriteAll() errors under strict confinement
```

```
    because the bundled snap cannot exec host xclip / wl-copy binaries; the
    handler discards the error with `_ =`.
  Reasoning mode: deduction.  Credence: 0.95.
  Status: WITNESSED (root cause confirmed against code + snapcraft + maintainer
    thread).

H1: switch snapcraft confinement: strict → classic
  Status: KILLED. Snap Store manual review required; not code-only.
H2: grant desktop / x11 / wayland plugs
  Status: KILLED. plugs grant socket access but do not bundle the host
    binaries the upstream clipboard library shells out to.
H3: bundle xclip / wl-clipboard inside the snap
  Status: KILLED. out of scope per maintainer thread.
H4..H6: README warning / fx.wtf docs / native Go clipboard backend
  Status: KILLED on venue, repo boundary, and library-swap scope respectively.

[Re-entry: prior pass froze at "docs PR" verdict, but the frontier was not
 closed: every leaf had been killed by scope or venue rather than evidence
 against the underlying mechanism.  Audit routed back to recon.]

H7: OSC 52 escape sequence as additive write path
  Perturbation surface: copyToClipboard() wrapper; transitive dep
    github.com/aymanbagabas/go-osc52/v2 promoted to direct.
  Falsifiable predicate: terminal-emulator OSC 52 sequence
    (\x1b]52;c;<base64>\a) emitted to os.Stderr (fx's TUI render channel)
    sets the system clipboard with no external binary and no snap plug.
  Reasoning mode: abduction (option) → deduction (stderr channel, write-only
    use) → induction (build passes; sequence verified).  Credence: 0.80.
  Status: WITNESSED. `go build ./...` passes; sequence correct.  Runtime
    confirmation in a confined terminal pending.
```

Three structural elements are visible in the node-set: typed reasoning-mode labels (`deduction` on H0; the abduction-then-deduction-then-induction trace on H7), stakes-indexed credence (0.95 on the root cause, 0.80 on the proposed fix pending runtime confirmation), and mechanical kill predicates (H1 by Snap Store policy; H2 by the missing-binary deduction; H3–H6 by scope and venue). The H7 discovery on re-entry is the loop's intended behavior: the earlier verdict was structurally premature because H1–H6 closed by scope rather than by evidence against the underlying mechanism, leaving the frontier open. The deterministic gate routed the trajectory back to recon, which proposed H7, and audit witnessed it under a build check. The corpus of ~380 such files is the deployment trace referenced in §5.6 and §7.

0.4.3 3.3 Blind-blind pushout at the hypothesis stage

Two frontier models from different families receive the same evidence pack with no cross-visibility, and each produces a hypothesis independently. A third pass extracts the disagreements, not the agreements: the disagreement becomes the next node in the graph; the agreement is recorded but not actionable. Adversarial filtering operates at hypothesis time, while the worktree is still untouched, rather than at patch time where the diff is already written. Sampling stochasticity alone produces real divergence even within a single model; cross-family divergence compounds it with architectural and training-corpus differences. Both are signal.

0.4.4 3.4 Trajectory-shape classification

Each audit re-entry produces an outcome trace for the active hypotheses: which $FAIL_T O_P ASS$ tests flipped, which $PASS_T O_P ASS$ tests regressed, which related kill predicates fired. The evaluator labels the trace's shape across re-entries with one of four qualitative classes drawn from sequential-evidence and dynamical-systems lineage (Wald 1947; Vovk & Wang 2021; Ramdas 2023; Milnor 1985; Strogatz 2014).

- Convergent. Evidence trends toward witnessing one hypothesis across re-entries (the test responses progressively favor it; sibling hypotheses fade).
- Divergent. Evidence trends toward two or more sibling hypotheses simultaneously (the responses partition the test set rather than concentrating on one).
- Oscillatory. Evidence flips between hypotheses across re-entries without stabilizing (the same node alternates witness and kill across passes).
- Chaotic. No stable trend across re-entries.

The lineage is borrowed for the *taxonomy*, not as a deployed numerical accumulator: we do not implement a per-test-outcome multiplicative likelihood-ratio score with a fixed two-sided threshold in v1. A formal anytime-valid accumulator would require IID Bernoulli outcomes, which SWE-bench audits do not satisfy; the qualitative four-shape labels are what the gate consumes. The shape label is the deployed kill-decision input; the formal accumulator is design specification for a later version where the IID assumption can be earned (e.g., bootstrapped per-instance test resampling).

The deterministic gate (§3.5) reads the shape label emitted by the evaluator. Both the evaluator and the gate are mechanical; neither is a model call.

0.4.5 3.5 Deterministic gating

The gate is a finite-state classifier over (trajectory shape, attempt count, budget remaining, frontier-closure status), not a model call. Outputs are `continue-into-craft`, `re-enter-recon-with-graph-update`, `terminate-success`, `terminate-budget-exhausted`, and `escalate-to-human` (private set only). No stage may decide its own termination; the gate decides. The gate's logic is published, reviewable, and frozen by the same tag as the rest of the artifact. Termination is mechanical, which is what makes the run reproducible: re-running the same evidence trace through the same gate yields the same routing.

0.4.6 3.6 Outer-loop iteration

Audit failures do not retry the patch. They route back to recon with the trajectory classification and the updated graph. Craft sees a different node-set on re-entry rather than the same node twice. The attempt budget per instance is bounded. Budget exhaustion counts as a clean termination without convergence, and the loop treats it as a verdict. No peek at held-out tests at any point in the loop. The gate's inputs are local to the instance and the agent's own run.

0.4.7 3.7 Fault classification (operator-side)

Pre-registered fault classes cover environment-induced failures: auth outage, quota exhaustion, infra timeouts. These are platform faults, distinct from reasoning losses. Classification triggers on fault-window membership, not on instance verdict. Wins inside the window get re-run alongside losses. Any reclassification rule that lets a losing run be retried while letting a winning run stand is disallowed; asymmetry is exactly the loss-laundering the protocol forbids.

0.4.8 3.8 One-shot held-out discipline

The public set may be iterated through the outer loop and re-run on platform fault. The held-out set is graded once on a frozen artifact, with no per-instance feedback consumed as an iteration signal; the held-out grade is treated as an oracle, never a stopping signal. The artifact that earns the submission hash is the submission. Local-green/official-red is structurally impossible because the hash binds the captured prediction to the run that earned it.

0.4.9 3.9 Preregistration and freeze

The artifact is frozen by an annotated git tag (`prereg-pro-v1`); every scored-run artifact cites the freeze SHA. A new tag opens a new worklog with the failure class that justified the restart named explicitly. v1 is the first scored tag.

0.4.10 3.10 Provenance contract

Per-instance trajectories (agent sessions for each pipeline stage) are captured off-box on a polling cadence. A hypothesis graph document, captured diff, grader output, and gate trace are committed per instance, alongside a per-box ledger and cost provenance. A run earns headline status only when full provenance is published; scores alone are insufficient.

0.5 4. Experimental Setup

0.5.1 4.1 Datasets and eligibility

Two benches run under one frozen loop. SWE-bench Verified is a curated subset, saturated (top public submissions resolve >85% as of mid-2026) and training-cutoff-contaminated for current frontier models; we use it as a baseline that the loop ports cleanly, with companion repo `swebench-verified` and a frozen artifact under Zenodo DOI. SWE-bench Pro is the live contamination-resistant tier with a public/held-out split across different repos, and it is the primary validation surface. Both runs use the official harness; no bespoke graders. Defect lists are documented per bench (`KNOWN_BAD.md`) and cover bench defects only, never our failures; the eligible set is the full set minus documented defects.

The two-bench design dissociates bench saturation (Verified) from loop generality (Pro). Same loop, two contamination regimes, two independent provenance trails. If the loop carries from one to the other at comparable per-repo rates, the assembly is not bench-specific.

0.5.2 4.2 Execution

Fixed run order, no early stopping. Whole-set evaluation on a multi-box fleet with per-instance isolation. Dynamic coordination with fault-tolerant resume across box reprovisioning.

0.5.3 4.3 Models and roles

No training, no fine-tuning, no reinforcement learning, no in-context demonstrations from prior instances. Frontier models are queried via API at their published checkpoints; the harness contains no learned weights of its own. Generator and challenger are drawn from current frontier families, deliberately cross-family to avoid mode collapse on a single training prior. Models are pluggable: the inquiry loop and smem operate over any pair of capable frontier models with structured-output and tool-use support. We freeze the specific model versions used for this run's reproducibility, but no part of the methodology is specific to a particular model. Auth source is not part of the frozen artifact; billing mechanism is orthogonal to evaluation discipline.

0.5.4 4.4 Grading

Official harness only; no bespoke graders. Bench defects are noted in one line per instance; no per-instance forensic analysis that risks becoming bench-specific tuning.

0.5.5 4.5a Model-agnostic ablation (planned, pre-publication)

Before publication, the same frozen harness is rerun with Cursor Composer 2.5 as both generator and challenger, replacing the Sonnet + GPT-5.5 cross-family pair. SWE-rebench publishes Composer 2.5 at ~$0.23/problem, an order of magnitude below this work's blended frontier cost, projecting ~$170 to grade both benches end-to-end.

Three outcome readings are planned. If the result holds at a comparable rate, the typed contracts and the kill-conditioned graph are doing the work and model selection is not the lever; this closes the "model selection is unablated hyperparameter" critique. If the result degrades modestly, the loop is necessary but not sufficient, and frontier capability matters on the hard-tail repos (sympy, matplotlib, django) where Composer 2.5 falls short. If the result collapses, frontier capability is the dominant factor and the loop's contribution is negligible. The answer is published regardless.

This is a robustness run, not a clean isolation. The single-model run changes model family, capability, training cutoff, cost-per-instance, and cross-family disagreement simultaneously. The question it answers is whether the harness still produces results under a 6-7× cheaper model pair (Composer 2.5 at ~$0.23/instance per SWE-rebench, against Sonnet+codex blended at ~$1.50/instance); single-factor attribution is out of reach here. Clean per-factor ablations (typed-mode contract on/off, blind-blind pushout on/off, gate determinism on/off) remain future work and are scoped in §9. Composer 2.5 also has a different training cutoff than Sonnet and GPT-5.5; comparable performance across model families is informal evidence the methodology is not riding on the specific pretraining of any one model.

0.5.6 4.5 Cost and replication envelope

Two replication modes are validated in this work. *Subscription mode*: a single frontier-vendor consumer subscription (~$200/month) covers the run at a multi-week pace, gated by per-window quota. Marginal cost at the boundary is zero; time cost is weeks. Used for the early portion of this run before the API switch. *API mode*: pay-as-you-go API access at the published rate, validated cost approximately $1.50 per instance blended across light and heavy repos, for a total run cost in the ~$1k range across both benches. Faster, capacity-bounded by the fleet's box count rather than the quota window.

Compute requirements outside the model are minimal. The harness runs on commodity Linux boxes (we used 4× small EC2 instances during the API window): no GPU, no curated training data, no offline pipeline. Per-instance dollars and wall-time are committed to the ledger alongside the verdict and the trajectory, so replicators can audit what the run cost in real currency.

A consumer subscription or a sub-$1k API budget puts replication within reach of graduate students, independent researchers, and small teams; no special funding required.

0.6 5. What the paper reports

> No statistical apparatus. Per §3.1, each instance is a deterministic causal experiment; a single passing test on a captured diff is a complete verdict for that instance. The paper reports counts, denominators, and per-repo breakdowns, and leaves interpretation to the reader.

0.6.1 5.1 What the tables contain (per-bench)

Counts of WIN, LOSS, and INCOMPLETE for each bench, with denominator (eligible set = full set minus documented `KNOWN_BAD.md` defects at the freeze SHA). Per-repo rows record repo, instance count, WIN, LOSS, INCOMPLETE, mean cost per instance, and median wall-time. Aggregating across repos is a weighted average over uneven coverage; we report the per-repo table as the honest read and let aggregation be a derived computation the reader can do. The cost ledger commits per-instance dollars and wall-time alongside the trajectory; replicators can audit the total run cost down to the instance. Hypothesis-graph metrics per terminal instance include graph depth at termination, frontier-closure status (closed or budget-exhausted), count of kill-conditions fired, and count of nodes added across all outer-loop iterations. The gate-routing distribution reports the frequency of each gate output (`continue-into-craft`, `re-enter-recon`, `terminate-success`, `terminate-budget-exhausted`); the deterministic gate's behavior is itself a reportable property.

0.6.2 5.2 Treatment of incomplete and faulted instances

Fault classes (§3.7) are `AUTH_OUTAGE`, `QUOTA_EXHAUSTED`, `INFRA_TIMEOUT`, `CRAFT_HANG`. Classification triggers on temporal-window membership regardless of instance verdict; wins inside the window get re-run alongside losses, and asymmetric re-runs are forbidden. Each faulted instance gets one re-run; terminal verdicts stand, and faulted-again rows remain `INCOMPLETE` in the final ledger. Out-of-window verdicts stand: no reclassification of a real-output verdict on grounds of post-hoc inspection. `INCOMPLETE` gets its own column alongside `WIN` and `LOSS` rather than being folded into a denominator-mangling adjustment, so the reader sees the bounded honest cost of the run.

0.6.3 5.3 Two-bench reading

For each repo present in both eligible sets, we report the per-repo Verified count and Pro count with denominators. Both numbers per repo are presented side by side, and whether the assembly carries is left to the reader's interpretation. We do not report Wilson CIs, paired tests, or equivalence procedures. The per-instance verdicts are deterministic; aggregation is bookkeeping. A reader who wants confidence intervals can compute them from the published counts.

0.6.4 5.4 Independent re-grade

Twenty random WINs from the union of both benches, stratified by repo to avoid light-repo bias, drawn with a fixed random seed committed before the re-grade. The re-grade protocol applies the captured diff plus the official harness on a clean container, by an independent operator who has not seen the trajectory; it confirms or refutes the original WIN verdict per instance. The output is a `k / 20` confirmation count plus per-instance notes on the (20−k) non-confirmations if any. The confirmation count is compared against the published rate.

0.6.5 5.5 Composer-2.5 robustness run (§4.5a)

Same counts, same tables, computed independently for the Composer-2.5-only run. The Sonnet+codex per-instance verdict and the Composer-2.5 per-instance verdict are reported side by side for each instance in the intersection. The disagreement count is read directly; no statistical procedure applied.

0.6.6 5.6 Results

> *[Final tables inserted after v1 termination and §3.10 provenance publication. Current state is summarized below; per-repo and provenance tables expand in place once the run terminates.]*

Verified (closed, public). 426 / 438 eligible WIN under the official grader. Denominator-explicit; per-repo table committed to the `swebench-verified` repo's `results/` directory; Zenodo DOI pins the frozen artifact. The full 500-instance Sankey (eligible 438 = 500 − 44 sphinx-doc offline-infeasible − 18 documented `KNOWN_BAD.md` defects) is in the companion README.

Pro (in-flight, current state 2026-05-28). - Terminal grades: 196 WIN / 8 LOSS = 96.1% (N = 204). - INCOMPLETE: 198 (re-run queue draining; fault-window reclassifications applied per §3.7). - Unique instances seen: 402 / 728 eligible. Untouched: 326 (heavy tail of sympy, matplotlib, django, etc., pending). - 8 standing LOSSes: all real-diff "not resolved" under the official grader; none in a fault window. Distribution: protonmail/webclients (4), flipt-io/flipt (1), gravitational/teleport (2, early in run), 1 other. - The headline rate is provisional: the touched 402 are weighted toward early-order light repos. Final Pro number lands at run termination; provenance per §3.10 follows the same publication discipline as Verified.

Cost (Pro, in-flight ledger). ~$2 per audited instance (Sonnet + GPT-5.5 blended); ~$210 spent across the first ~140 API-mode instances at the time of writing; trajectory ~$1k for the full bench. Per-instance dollar ledger committed alongside trajectories.

OSS deployment trace. 80 merged PRs across 46 repositories at the time of writing, 53% merge rate under adversarial maintainer grading (GraphQL-verifiable via the queries pinned at `kimjune01/kimjune01@paper-2026-05-28/README.md`; profile README is volatile, the receipt is the pinned commit). 67% positive reception on 65 issues filed since the slop-filter campaign start. ~380 hypothesis graphs publicly committed in `sweep/repo-hypotheses/`.

0.7 6. Limitations

Public Pro is training-cutoff-contaminated for both model families used. This is a property of the bench, shared by all submitters, not an isolation of method as the cause. Self-holdout is weaker than Scale's: different commits, same repos. Cross-repo generalization to Pro's held-out partition is untested locally. Training-cutoff math for both models is published explicitly, with per-instance date overlaps disclosed.

One-shot held-out discipline is verifiable only at submission time. The submitter-side invariant (no held-out feedback in the artifact) is necessary but not sufficient. The run is cost-bounded; heavy repos may be under-represented or over-budgeted depending on order, with cost and time logged per instance.

Selection of generator and challenger families is a hyperparameter we do not ablate. The ablation that would isolate the loop's effect (with vs without the typed-mode constraint, on one fixed model) is a separate clean-room study. Because no training or fine-tuning occurs in this work, pretraining-cutoff contamination is the only contamination channel; reviewers concerned about it need only audit the base models' published cutoffs against the bench's instance dates.

The smem is small. Hypothesis graphs in this work are per-instance; cross-instance memory accumulation is future work. The smem proposal is therefore tested in its simplest non-trivial regime.

0.8 7. Discussion

> Verified is closed: 426 / 438 eligible, frozen, Zenodo-DOI'd, public. Pro is in-flight: current state 2026-05-28 is 196 WIN / 8 LOSS = 96.1% on 204 terminal grades (402/728 unique instances touched, 326 untouched). The Pro number is provisional pending run termination; Pro-specific readings below are marked accordingly. Discussion of any deviations once Pro terminates is written after the v1 run.

The loop ports cleanly to industrial code at the controlled-bench tier. 97% of eligible Verified instances resolved under the official grader with full per-instance provenance: captured diffs, trajectories, and cost ledger committed. The number is public and reproducible. The hypothesis-graph smem records typed inquiry decisions per run (kill conditions fired, evidence trajectories classified); whether those records prove reusable across instances and across benches is the empirical question the Pro run will speak to. The deterministic gate produces reviewable termination traces: an external auditor can replay its decision against the captured trajectory and budget, and the routing is reconstructible from the evidence trace alone.

Reproducibility comes from the captured trace, the frozen artifact, the deterministic gate, and the qualitative shape labels feeding it, not from Peirce. Peirce names the contract vocabulary and makes it legible; the mechanical enforcement (stage contracts that reject mode-freelancing inputs, captured trajectories with hypothesis-graph snapshots, finite-state gate logic, deterministic shape-to-routing mapping) is what lets the run replay.

The composition was designed for principled reasons. The hypothesis graph's shape (Peirce-typed nodes, e-value-inspired kill-conditioned edges, one-mode-per-stage write contracts, model-free gates) was designed to close known failure modes of LLM agents: mode collapse, confabulation, unauditable revision, vibes-based termination. The design rationale is independently inspectable from the artifact, and Verified shows the design clears industrial code at a 97% eligible rate. Whether Pro replicates that depends on the in-flight run. Conditional on Pro completing: if the per-repo Pro rate approximately matches Verified at the repos in both eligible sets, the assembly is bench-agnostic and contamination-regime-agnostic at the per-repo level; if Pro lands materially lower, the asymmetry between the two contamination regimes is what the reported gap means.

The OSS PR record (80 merged across 46 repos, 53% merge rate, adversarial maintainer graders; GraphQL verifiers pinned at `kimjune01/kimjune01@paper-2026-05-28/README.md`, since the profile README itself drifts) reports what the harness does on contact with non-curated codebases. The ~380 publicly committed hypothesis graphs at `kimjune01/sweep/repo-hypotheses/` are the deployment trace: skip verdicts, dead-end paths, engage decisions. This is ecological, not a controlled ablation; the deployment surface lacks a within-instance comparator, and isolating the typed-mode constraint requires the planned Composer-2.5 robustness run (§4.5a). The merge rate is well above the AI-slop floor visible in the same pinned commit's slop table for contemporaneous unstructured attempts.

Portability follows from the no-training stance. The harness contains no learned weights of its own; anyone with frontier-model API access can clone the repo, swap in their model pair, and run the loop without training compute, curated datasets, or GPU resources. Weights are pluggable; the cost envelope (§4.5) fits inside an individual researcher's discretionary spend.

One peripheral note on bench infrastructure: held-out validity is partly a function of who gets graded. Published

access rubrics (the way submission formats are published) would strengthen Pro's claim to be community infrastructure. We offer this as a structural observation.

0.9 8. Related Work

0.9.1 8.1 SWE-bench, contamination, and cost transparency

The SWE-bench family defines the Verified / Pro lineage, official harness, and contamination-resistant tier design. SWE-rebench (swe-rebench.com) uses post-cutoff filtering as a parallel contamination strategy and publishes per-problem cost figures (e.g., Cursor Composer 2.5 at $0.23/problem), used here as the price anchor for the model-agnostic robustness run in §4.5a. Adaptive data analysis (Dwork et al.) provides formal grounding for held-out-budget discipline. The official SWE-bench experiments repo (github.com/swe-bench/experiments) requires `trajs/`, `logs/`, `patch.diff`, `report.json`, and `test_output.txt` per submitted instance, establishing the minimum publication norm; our provenance contract extends it with gate traces, hypothesis graphs, and cost ledger. The Nilenso SWE-bench Pro trajectory analysis (nilenso.github.io/swe-bench-pro-cost-token-time-analysis) reports deterministic intent classification, exit statuses, structural markers, and cost/error signals across four frontier models on Pro; closest precedent on Pro cost transparency, not a per-instance dollar ledger. The Datacurve DeepSWE leaderboard reports median per-trial cost for frontier models on Pro-class instances (e.g., GPT-5.5 at ~$5.80/trial median); journalism-level cost data, not a structured ledger.

0.9.2 8.2 Agent scaffolds, embodied loops, and SE-agent harnesses

SWE-bench-targeted agent harnesses include OpenHands (Wang et al. 2024), SWE-agent (Yang et al. 2024), and AutoCodeRover (Zhang et al. 2024/25). These are methodological neighbors with different scaffold trade-offs; none implements Peirce-typed stage contracts or a kill-conditioned hypothesis-graph memory. Voyager (Wang et al. 2023) is the closest loop-shape precedent: embodied observe→hypothesize→test→commit in Minecraft. We adopt the loop shape, type its stages with Peircean modes, substitute kill-conditioned hypothesis graph for skill library (falsifiable belief in place of verified code), and re-substrate to industrial code. Invariant Risk Minimization (Arjovsky et al. 2019) is tri-abductive figure-ground splitting under environment variation; a third witness to the loop pattern from distributional generalization. SWE-Replay (2026) evaluates a test-time scaling method across Verified, Pro, and Multilingual; adjacent on cross-bench evaluation, not on frozen-artifact preregistration.

0.9.3 8.2a Peirce-typed reasoning for LLM agents (parallel work)

IDEA (He et al. 2025, ACL Findings, arXiv:2408.10455), *Enhancing the Rule Learning Ability of Large Language Model Agent through Induction, Deduction, and Abduction*, explicitly cites Peirce and uses the three modes in an interactive LLM-agent rule-learning benchmark; not SE-targeted, no persistent hypothesis-graph memory, no deterministic gate. ADI (Gilda & Gilda 2026, arXiv:2604.15727, April 17 2026), *Structured Abductive-Deductive-Inductive Reasoning for LLMs via Algebraic Invariants*, gives an explicit Peircean tripartite protocol with epistemic layers (L0/L1/L2) and a symbolic knowledge graph; near-simultaneous with this paper's draft and the most conceptually adjacent prior work. ADI targets algebraic-invariant reasoning; this paper targets SE agents on real industrial code under benchmark and adversarial-maintainer evaluation.

0.9.4 8.2b Graph-structured memory and typed-belief representations for LLM agents

The hypothesis-graph assembly sits at the intersection of three lineages: cognitive-architecture memory (Soar / ACT-R / EPIC), LLM-agent memory systems (CoALA / AriGraph / Mem0 / Zep), and typed-belief / falsifiability representations (CausaLab / BeliefMem / Theorem-of-Thought / CMM). Adjacent work in each lineage:

- Soar / ACT-R / EPIC (Laird 1987 forward; Anderson; Meyer & Kieras): the cognitive-architecture memory tradition. Working/declarative/procedural/preference memories; chunks; mechanical production-rule operations. Pre-LLM read/write semantics; the lineage establishes that *structured agent memory with mechanical operations* is decades old.

- Kirk, Wray & Laird 2023 (AAAI): Soar/ITL agent queries an LLM, verifies, encodes into memory, uses chunking. The LLM-port of Soar lineage; internal representation remains Soar-shaped, not LLM-prose-shaped.
- Soar's memory formalization (Laird and the Soar manual; the Soar architecture's distinction between semantic memory `smem`, procedural memory `pmem`, and episodic memory `epmem`). We adopt this three-way memory typology directly: the hypothesis graph is our `smem`; the pipeline-stage skills are our `pmem`; the captured per-run trajectories are our `epmem`. The categorization is Soar's, not ours; what we add is the specific content filling Soar's `smem` slot: a Peirce-typed kill-conditioned graph designed for LLM read/write semantics.
- CoALA (Sumers et al. 2023/24, TMLR, arXiv:2309.02427): cognitive-architectures framework for language agents with modular memory. Broad foundation.
- AriGraph (Anokhin et al. 2024/25, IJCAI, arXiv:2407.04363): knowledge-graph world models for LLM agents in TextWorld. Closest precedent for *graph-structured* LLM agent memory; nodes are entities/relations/episodes, not falsifiable hypotheses.
- CausaLab (Yang et al. 2026, arXiv:2605.26029): LLM agents maintain evolving structural-causal-model hypotheses in a DSL. Strong adjacent on *persistent inspectable hypothesis representation*; causal-discovery domain.
- BeliefMem (Liao et al. 2026, arXiv:2605.05583): multiple candidate conclusions with non-LLM probabilistic Noisy-OR updates under partial observability. Strong adjacent on *uncertain alternatives + mechanical update*.
- Theorem-of-Thought (Abdaljalil et al. 2025, arXiv:2506.07106): abductive / deductive / inductive *agents* produce traces structured into formal reasoning graphs; NLI-guided Bayesian coherence scoring. Strongest Peirce-mode adjacent; modes operate at agent/trace level, not as falsifiable hypothesis node types in persistent memory.
- CMM (Khalid & Arora 2026, *From Observed Reasoning to Stable Skills*, Agent Skills '26, published 2026-05-26 (one day before this draft), OpenReview): Cognitive Memory Manager: a memory substrate that captures coding-agent execution and extracts reasoning structure as a typed DAG. Seven node types (HYPOTHESIS / INVESTIGATION / DISCOVERY / PIVOT / DEAD_END / SOLUTION / CONTEXT_LOAD), edges linking hypotheses to investigations and dead ends to pivots and solutions, two-tier extraction (warm at session-stop hook, cold via LLM later), human-approval gate, retrieval-validated graduation into stable `SKILL.md` files at default thresholds (3 distinct sessions for project-scoped patterns, 2 for team-scoped). The published case-study evaluation ingests sessions from one developer over six months. The closest contemporary precedent on typed-DAG coding-agent memory, independently developed and published 24 hours before this draft.

  Convergent on a typed persistent reasoning-memory role; categorically different runtimes. Both systems converge on a similar role for memory: a persistent, typed, queryable DAG of reasoning artifacts that other components read and write. (We describe our slot using Soar's `smem` terminology; CMM does not adopt Soar's vocabulary, so the slot-mapping is our framing applied to a comparison, not an attribution to their architecture.) The data structures look similar at the interface level, as does the hypothesis / investigation / dead-end vocabulary.

  The runtimes differ in the actor's role. CMM's runtime is *observe-and-consume*: an external coding agent does the perturbing, CMM captures the resulting trajectory and types it post-trajectory, and later runs consume the graduated skills via tools like `diagnose`, `pitfalls`, `search-memory`. Our runtime is *perturb-and-falsify*: each craft step is the agent itself applying a candidate patch, each audit step is the observable test response, and kill conditions fire mechanically on evidence accumulation during the live inquiry. CMM extracts, consolidates, and serves memory to future work; our harness uses memory as the live substrate for selecting patches, running tests, firing kills, and routing. That is a categorical asymmetry of agency, not of rigor.

  The categorical asymmetry refracts into five axes:

- (i) Prior requirement. CMM's demonstrated advantage depends on relevant prior session memory; the case-study evaluation runs on six months of one developer's sessions, and graduation requires at minimum 3 distinct sessions per project-scoped pattern (configurable, default). The system can begin collecting after one run, but the demonstrated benefit accrues with empirical session memory. Our inquiry loop operates on previously-unseen instances cold, with zero task-specific session history and zero per-developer tuning. CMM's output is a personalized profile distilled from prior; our output is a graded fix on first contact.
- (ii) When the typing fires. CMM applies types at *extraction time*: post-trajectory rather than during the agent's reasoning. The warm tier runs at the session-stop hook; the cold tier does LLM extraction later. Those types are not inert labels; they shape downstream clustering, consolidation into pitfalls/insights/diagnostic strategies, and consumption by future agents through tools like `diagnose`, `pitfalls`, and `search_memory`. So CMM's types are runtime-active *at the consumption layer*. Our Peirce-mode types are runtime-active *at the write layer*: each pipeline stage's writes are mechanically restricted to its mode, rejecting mode-freelancing inputs at the moment they would be produced. CMM types describe and shape; ours constrain. CMM's seven types vaguely match the three Peircean modes at the surface (HYPOTHESIS / INVESTIGATION / DEAD_END / SOLUTION map loosely to abductive / inductive-evidence / inductive-kill / inductive-witness), but the locus of enforcement is different.
- (iii) Edge semantics. CMM uses causal relationships and confidence decay; we use mechanical kill-condition predicates over evidence.
- (iv) Write contracts. CMM's agent writes any node type from a single stage; we constrain each pipeline stage to one Peircean mode.
- (v) Temporal scope. CMM accumulates across months of sessions to graduate stable Skills; we reset per-instance and use the graph as a per-problem inquiry trace under deterministic finite-state termination.

Perturbation distinction (precise version). CMM captures perturbations performed by agents: INVESTIGATION nodes record file reads and command executions; DEAD_END nodes record errors, test failures, and futile commands. The evidence model preserves causal arcs from hypotheses through investigations to dead-ends, pivots, and solutions. What CMM does *not* do, and what the inquiry loop in this work does, is itself select perturbations, apply patches, run falsifying tests, or enforce kill conditions during the live inquiry. Those choices belong to the agent CMM is observing.

Our inquiry loop is the actor: craft chooses the patch, audit runs the test and classifies the trajectory shape, the gate routes on the shape label, and the kill predicate on the node fires mechanically. Both systems engage with perturbative evidence; only one *is* the perturbing actor.

Where abduction and induction sit, in each system. Within an observed transcript, a CMM HYPOTHESIS node is a faithful post-hoc label on an abductive move the underlying agent made; *"maybe the issue is in the routing factory"* is an abductive proposal whatever system labels it. That labeling is honest. CMM's *graduation pipeline*, however, is inductive: recurring HYPOTHESIS-bearing patterns across sessions are clustered, validated by retrieval, ratified at threshold, and promoted into procedural advice. The graduated product (the `SKILL.md`) is induction-from-cases. So: CMM's individual-node typing is observational and Peirce-compatible; CMM's consolidated output is inductive by construction. Our pipeline splits the modes by stage so the operation a stage may perform is constrained at write time, regardless of what the underlying agent does.

Bibliographic note. CMM's references sit in the agent-memory and long-context-LLM literature (Mem0, Letta, Cognee, ExpeL, Reflexion, Auto Memory, cursorrules, Lost-in-the-Middle, Chroma's long-context study). The paper uses hypothesis/investigation terminology that is compatible with Peircean inquiry but does not claim Peircean grounding, and the bibliography does not engage the Peirce, Ramsey, or Wald [?] Vovk-Wang [?] Ramdas lineage. This is consistent with the broader field pattern in §2: adjacent systems on the loop arrive at the vocabulary empirically, without naming the philosophy-of-science substrate.

The two systems are *complementary by construction*: our ~380 publicly committed hypothesis graphs are exactly the kind of corpus CMM's graduation algorithm could consume to extract specialized per-repo skills

downstream. A future system could chain them: inquiry loop (this work) perturbs code and produces structured inquiry traces across many instances; personal skill-graduation algorithm (CMM) consolidates those traces into a developer-or-repo-specific Skill set.

Production LLM memory systems with graph variants (Zep/Graphiti, Mem0), staged-hypothesis selection in science agents (DeepScientist), domain-specific hypothesis swarms (drug discovery), deterministic gating in adjacent settings (MemLineage, ROE Gate, Certifying-Risks, FVA-RAG, GHS-TDA), and reflective memory systems (Reflexion, DebugMate, Potpie) are surveyed in the appendix. They are adjacent in particular axes but do not change the cell-by-cell comparison spine above.

0.9.5 8.2c Multi-model adversarial filtering

- Multi-Agent Debate (Liang et al. 2023/24, EMNLP, arXiv:2305.19118): foundational adversarial multi-agent disagreement framing; operates with cross-visibility at patch stage in non-SE domains, where this work runs blind at hypothesis stage on SE.
- Refute-or-Promote (Agarwal 2026, arXiv:2604.19049): adversarial stage-gated multi-agent review with cross-model critique and context asymmetry, for high-precision defect discovery. Strong adjacent for *blind multi-model filtering*; operates at defect-finding/review stage, not pre-patch hypothesis stage.

0.9.6 8.2d Agent termination and trajectory analysis

- λ_A: A Typed Lambda Calculus for LLM Agent Composition (2026, arXiv:2604.11767): proves termination of bounded fixpoints for agent composition. Formal termination, not evidence-shape gating.
- SafetyDrift (2026, arXiv:2603.27148): models agent safety trajectories as absorbing Markov chains for risk analysis. Trajectory-level stopping, not convergent/divergent/oscillatory/chaotic evidence bins.

0.9.7 8.3 Peircean inquiry and the philosophy of science

- Peirce 1878 (*Illustrations of the Logic of Science*, Popular Science Monthly): the original three-mode taxonomy of abduction, deduction, and induction.
- Peirce 1903 (*Pragmatism as the Logic of Abduction*, Harvard lectures): abduction as the only mode that introduces new content.
- Bacon 1620 (*Novum Organum*): induction as a typed primitive.
- Popper 1934 (*The Logic of Scientific Discovery*): falsification as the inductive-side constraint.
- Ramsey 1926 (*Truth and Probability*; in *The Foundations of Mathematics and other Logical Essays*, 1931): operational credence as betting odds, the Dutch Book argument, subjective probability as the substrate for graded belief. The hypothesis graph's node-level semantics descends from this work.
- James 1907 (*Pragmatism: A New Name for Some Old Ways of Thinking*) and Dewey 1929 (*The Quest for Certainty*): pragmatist commitment that truth is inseparable from action; the stakes-indexing of belief follows from this commitment.
- Meehl 1967: soft-science methodological critique that anticipates many of the failure modes our typing constrains.
- Feynman 1974 ("Cargo Cult Science"): informal but substantive on the difference between rigor-shaped activity and actual rigor.

0.9.8 8.4 Bi-abductive and compositional inference; failure isolation

- Calcagno et al. 2009: compositional shape analysis via bi-abduction; Facebook Infer as the industrial-scale instance of typed-mode inference on real code.
- Bylander et al. 1991: abductive computational complexity.
- O'Hearn 2019: separation logic and incorrectness logic, the modern compositional-inference scaffolding.
- Noam Zilberstein, Saliling & Silva 2024 (arXiv:2305.04842): Outcome Separation Logic; Theorem 5.1 establishes soundness of tri-abduction for branch composition under effects, extending bi-abduction from sequential to branching.

- Zeller & Hildebrandt 2002 (*Simplifying and Isolating Failure-Inducing Input*, IEEE TSE; building on Zeller 1999): delta debugging. An optimization-side adjacent: given a failure-inducing input, isolate the minimal failure-inducing subset by binary-search-shaped perturbation. The hypothesis graph in this work is methodology-shape; delta debugging is optimization-shape; both rely on the reproducible / deterministic / perturbable properties of code (§3.1). Worth citing as the canonical demonstration that mechanical perturbation of code is a productive inference primitive.

0.9.9 8.5 Anytime-valid inference and evidence trajectories

- de Finetti 1937: subjective probability foundations.
- Ville 1939: martingales as the formal substrate for sequential evidence.
- Wald 1947: sequential testing of statistical hypotheses; the original "evidence is time-indexed" insight.
- Robbins 1967; Chernoff 1959: sequential design and stopping rules.
- Kelly 1996: capital growth and the betting-theory tie to e-values.
- Vovk & Wang 2021 (*E-values: Calibration, combination, and applications*, Annals of Statistics): the framework for e-values as non-negative random variables with $E_{H_0}[E] \leq 1$, calibration of p-values to e-values, combination, and anytime-valid stopping by Ville's inequality. The temporal-evidence stance in this paper (§3.4) is *inspired by* the sequential-evidence-trajectory framing; the four-shape taxonomy descends from this lineage even though v1 does not deploy a formal e-value accumulator (the IID assumption SWE-bench audits violate would be required).
- Shafer 2021: the *betting* interpretation of e-values; an e-value as the wealth of a strategy that bets against the null.
- Grünwald 2024; Ramdas 2023: modern syntheses of anytime-valid inference, e-processes, and safe testing.

0.9.10 8.6 Dynamical classification

- Milnor 1985; Strogatz 2014: dynamical systems literature for convergent / divergent / oscillatory / chaotic shape classification.

0.9.11 8.7 Directed acyclic graphs as reasoning representation

- Pearl 1988 (*Probabilistic Reasoning in Intelligent Systems*): Bayesian networks as DAGs of probabilistic dependencies. The foundational application of DAGs to structured belief representation.
- Pearl 2000/2009 (*Causality: Models, Reasoning, and Inference*): structural causal models, d-separation, do-calculus. DAGs as the substrate for causal-structure inference.
- The hypothesis graph in this work uses Pearl's DAG representation primitive but for a different purpose: typed *hypotheses about engineered systems under inquiry*, not probabilistic dependencies between random variables or causal relations between exposures and outcomes. The data structure is borrowed; the semantics over it are not.

0.10 9. Future Work

Four directions follow from this work. A held-out submission under the same artifact, one-shot discipline. Cross-instance smem accumulation: letting the hypothesis graph grow across instances within a repo, then across repos within a domain; the current work tests the smem at per-instance scope. A clean-room ablation on post-cutoff instances (SWE-rebench), with vs without the typed-mode constraint on one fixed model, to isolate the loop's effect on the rate. Skill-level retros: which stages of the loop carry which kinds of wins, and targeted skill freezes for follow-on benches.

0.11 10. Availability and Reproducibility

- Repositories. github.com/kimjune01/swebench-pro and github.com/kimjune01/swebench-verified.
- Frozen artifact. Git tag `prereg-pro-v1`, SHA committed in the worklog.
- Preregistration. `PREREGISTRATION.md` at the freeze SHA.

- Provenance artifacts. Per-instance trajectories, hypothesis graphs, captured diffs, gate traces, and cost ledger under `runs/scored/artifacts/`.
- OSS deployment trace. Public corpus of ~380 hypothesis graphs at `kimjune01/sweep/repo-hypotheses/`, one per investigated issue across 46+ repositories. PR-level outcomes (merged / closed-unmerged / open) are pinned at `kimjune01/kimjune01@paper-2026-05-28` (the profile README itself drifts; the pinned commit is the citable artifact).
- OSS metrics are recomputable, not asserted. Every numeric claim in the deployment trace (merge rate, repo count, issue reception) ships alongside the GitHub GraphQL query that produces it. A skeptic with any GitHub token can paste the query into the GraphQL explorer and recompute the number in under a minute. No statistic is presented without its verifier. This is the same anti-grift mechanism as the bench's per-instance provenance, ported to the wild-deployment surface.
- Replication budget. Sub-$1k of frontier-API spend, or a single consumer subscription plus patience. See §4.5.
- Companion textbook. A reader-facing synthesis of the same dispersed lineage is available at june.kim/reading/methodeutics. The paper's argument rests on the primary sources cited in §2 and §8, not on the synthesis; the textbook is a navigation aid, not authority.
- Public-provenance trail. Dated blog posts at june.kim establishing the framework: *Theory is load-bearing* (2026-03-17), *The proof manual* (2026-04-05), *Type the question* (2026-04-08), *Evidence has a trajectory* (2026-04-27), *The hypothesis graph* (2026-04-28), *Abduction* (2026-05-04), *Modes of reason* (2026-05-04). These predate CMM (2026-05-26) and ADI (2026-04-17) and establish parallel rather than derivative development.
- PDF. Arxiv-shape build at /assets/inquiry-loop-paper.pdf. Rebuilt from the markdown source by `scripts/build-paper-pdf.sh`; the source is canonical.
- DOI. [placeholder: Zenodo paper-DOI distinct from the software-DOI.]
- License. Skills are released free and openly under CC-BY-SA-NS (see june.kim/cc-by-sa-ns). Repo-level terms in `LICENSE.md`. The harness, the skills, the trajectories, the hypothesis graphs, the gate logic: all freely available for inspection, replication, modification, and reuse under attribution + share-alike + no-spam terms. No paywall, no gated access, no enterprise tier; the harness an outsider clones is the same harness that produced the published numbers.

Reproducibility invitation. *Nullius in verba.* The repository, per-instance trajectories, hypothesis graphs, gate traces, captured diffs, and cost ledger are public. The harness runs end-to-end on a frontier-vendor subscription or under ~$1k of API spend (§4.5). Replication does not require institutional credentials or enterprise access. Doubts about specific instances, regrades, or methodological claims should be filed as issues against the repository; confirmed corrections fold into the next versioned artifact.

0.12 11. Search Methodology

- Why this section exists. Several claims in this paper take the form *"we found no prior X."* Such claims are only as honest as the search they rest on. This section publishes the queries so readers can re-run the search and either confirm the gap or find what we missed.
- Sources searched. Google Scholar; arXiv (cs.LG, cs.AI, cs.CL, cs.SE); ACL Anthology; OpenReview; GitHub code and repository search; public SWE-bench leaderboard and submission archives.
- Queries by claim.
  - Peircean SE-agent loop. *"Peirce" "LLM agent" abduction deduction induction software engineering*; *"Peircean" "LLM agents" abduction deduction induction*; *"abduction deduction induction" "LLM agent" "software engineering"*; *"Peirce" "SWE-bench" agent*; *"code agent" "abduction" "deduction" "induction" LLM*.
  - Hypothesis-graph agent memory. *"hypothesis graph" "LLM agent" memory*; *"belief graph" "LLM agent" memory hypothesis*; *"knowledge graph" "LLM agent" "hypothesis" "memory"*; *"AriGraph" "knowledge graph world models" "episodic memory" "LLM agents"*.
  - Blind multi-model hypothesis-stage filtering. *"multi-agent" "code" "hypothesis" "SWE-bench"*; *"multi-model" "code agent" "disagreement"*; *"blind" "multi-agent" "code review" LLM*; *"ensemble" "LLM agents" "software engineering" "SWE-bench"*.
  - Trajectory-shape termination gates. *"LLM agent" termination criteria trajectory*; *"finite state" "LLM*

agent” “termination”; *“sequential testing” “LLM agents” stopping rules*.
  - Full per-instance provenance on SWE-bench Pro. *SWE-bench Pro leaderboard submissions trajectories cost ledger*; *“SWE-bench Pro” “trajectory” “cost”*; *“SWE-bench Verified” “trajectories” “cost ledger”*; *site:github.com “swe-bench-pro” “trajs”*.
  - Two-bench validation under one frozen artifact. *“SWE-bench Verified” “SWE-bench Pro” “same” “scaffold”*; *“SWE-bench Pro” “Verified” “frozen” “artifact”*; *“SWE-bench Pro” “SWE-bench Verified” “cross-bench”*.
  - Sub-$1k Pro replication and per-instance cost ledger. *“SWE-bench Pro” “cost per instance”*; *“SWE-bench Pro” “cost ledger”*; *“SWE-bench” “cost-per-instance” “leaderboard”*; *“SWE-rebench” cost per problem*.
- Caveats. The search is best-effort and bounded by visible-web indexing; private industry work, in-preparation manuscripts, and non-indexed venues are not covered. Discoveries of overlapping prior work post-publication should be reported as issues against the repository for citation update.

0.12.1 11.1 Comparative search supporting the headline claim

The headline claim, *no method documented has proved a higher SWE-bench resolve rate at a lower audited per-instance cost with equivalent receipts*, requires a comparative search separate from the novelty search above. The queries below are scoped specifically to that claim. The bar for *equivalent receipts* is: published per-instance trajectories, captured diffs, gate or evaluator traces, cost ledger, and reproducible run conditions.

Sources searched. SWE-bench Verified leaderboard (github.com/swe-bench/experiments); SWE-bench Pro official page (scaleapi.github.io/SWE-bench_Pro-os/); SWE-rebench public reports (swe-rebench.com); Nilenso Pro trajectory analysis; OpenHands, SWE-agent, AutoCodeRover, Aider, Devin reports; venturebeat / techcrunch reporting on Pro/Datacurve / DeepSWE; arXiv recent submissions tagged SWE-bench; vendor blogs (Anthropic, OpenAI, Google) on agent benchmark performance.

Queries. - *“SWE-bench Verified” “per-instance” cost trajectories*; *“SWE-bench Pro” leaderboard “cost” “trajectories”*; *“SWE-bench” submission “cost ledger”*. - *“SWE-bench” “cost per instance” submission diff trajectory*; *“swebench” submission reproducible cost*. - *OSS PR merge rate LLM agent maintainer-graded benchmark*; *agent-produced pull request acceptance rate*. - *“SWE-bench Verified” 95% 96% 97% top resolve rate trajectories cost*; *Pro leaderboard top resolve rate cost*.

Candidate audit (against the receipt bar). Each top public submission or comparable report is checked for: published per-instance trajectories (T), captured diffs (D), evaluator/gate traces (G), per-instance cost ledger (C), reproducible frozen artifact (R), and resolve rate at or above the rate this paper reports on the same bench (Rate). Receipt-bar columns are *present* (Y), *partial* (~), or *absent* (—). A row that doesn't combine all six is not a refutation of the headline.

Submission / report

Bench

T

D

G

C

R

Rate □ ours

Notes

Official swebench/experiments repo (multiple top entries)

Verified

Y

Y

—

—

~

Various

Minimum publication norm: trajs/logs/patch.diff/report. No gate traces, no cost ledger.

Top vendor leaderboard entries (Claude Code, OpenHands, SWE-agent, AutoCodeRover)

Verified

—

—

—

Reported below 97%

Submissions report numbers; reproducible bundles and cost ledgers rarely published.

SWE-bench Pro official page (Scale)

Pro

—

—

—

N/A (curator)

Uncapped cost (250-turn limit). No per-instance cost ledger.

Nilenso Pro trajectory analysis

Pro

—

N/A (third-party)

Cost/token/time analysis across four frontier models. Not a submission.

Datacurve / DeepSWE reporting (VentureBeat)

Pro-class

—

—

—

~

—

N/A (journalism)

Reports e.g. GPT-5.5 ~$5.80/trial median. Above sub-$1k bound at scale. Journalism-level, not structured ledger.

SWE-rebench public reports

rebench

—

Y

~

Below ours

Strong cost transparency (Cursor Composer 2.5 at $0.23/problem); reported resolve rates below the rates this paper documents on Verified.

This work: Verified

Verified

Y

Y

Y

Y

Y

426 / 438 eligible (97.3%)

Companion repo swebench-verified; Zenodo DOI; gate traces and cost ledger committed.

This work: Pro

Pro

Y

Y

Y

Y

Y

in-flight 196/8 = 96.1% on N=204 (2026-05-28)

Same frozen harness; 402/728 unique touched, 326 untouched; final headline at run termination. Provenance published per §3.10.

Reading. No row above this paper's two rows combines all six receipt-bar columns and a resolve rate at or above the rates this paper documents on the same bench. The headline survives as long as that table reads this way.

Caveat. Top-line resolve rates above the receipt bar may exist in private submissions or in submissions whose receipt set we have not been able to verify; the headline is bounded by what we documented. A citation showing a stronger combined receipt is the cleanest refutation.

0.13 LLM Collaboration Disclosure

Per current ethics norms for AI-assisted scientific writing, the use of LLMs in this work is disclosed here in two distinct roles.

Subject of study. The harness under evaluation (§3, §4) uses frontier LLMs as the generator and challenger inside the inquiry loop; this is the paper's object of inquiry rather than a writing aid, and the specific model versions, billing mode, and provenance are fully disclosed in §4.3 and the published artifact.

Writing aid. This paper was drafted with assistance from Anthropic's Claude (Opus 4.7) for converting a human-authored bullet-point outline into prose, for editorial passes (sharpening, tightening, em-dash and negative-parallelism linting), and for surface-form copyediting. All technical claims, citations, numeric results, methodology design, and the structure of the argument originate with the human author; LLM assistance was confined to surface-form editing of human-authored content. No LLM acted as a peer reviewer, determined any of the paper's claims, or selected what to publish. Every paragraph was read and approved by the human author before commit.

0.14 Acknowledgments

-

0.15 Draft notes (not for paper)

- Title decision. *"The inquiry loop on SWE-bench Pro"* with subtitle *"Hypothesis graphs as agent semantic memory, grounded in Peircean methodeutics."* Mirrors the user's existing canon (*The hypothesis graph*, *The spec is a hypothesis*, *Theory is load-bearing*, *Auditing DeepSWE*); declarative copula register, "The X" definite-reified pattern, bench tag for findability, contribution in the subtitle.
- Target venue. arxiv cs.LG primary, cs.AI cross-list. Endorser candidates: Mori (cs.LG, former-student route, first ask), Neubig (cs.CL/LG, OpenHands-Slack-warm route, stronger second ask after draft develops).
- Length target. 8–12 pages standard arxiv preprint; appendices for full provenance schema, hypothesis-graph format, fault classification details, gate state machine.
- Layering discipline. Tip / engineered / framework / foundation. The paper is readable at every depth: bench reader gets §1+§4+§5, methodology reader adds §3, framework reader adds §2, Peirce reader follows the citations.
- Sections 6–7 are load-bearing rhetorically. §6 surfaces limitations more aggressively than any reviewer would; §7 demonstrates substantive engagement with the bench's content and a *small* note on infrastructure (structural observation, not a complaint).
- Numeric results section deferred. Do not insert until both runs complete and the v1 scores stabilize. Insert in §5 with per-repo breakdown first, aggregate second, two-bench comparison third.
- Pre-circulation TODOs.
  - One-shot replay tool described in §3.10 (artifact rsync daemon; cleanest UX).
  - Independent third-party re-grade on a sample of WINs, cited in §5.
  - KNOWN_BAD run completed and disclosed.
  - The "legible tip of a longer lineage" sentence from the chat folded into §1 verbatim if it survives the next read.
- Distribution strategy.
  - arxiv preprint (with Mori endorsement for cs.LG).
  - OpenHands Slack as the warm channel where the work already has acknowledgment.
  - HN post with methodology-framing title (*not* the headline number) once the preprint has a DOI.
  - Short, narrow outreach to specific researchers (not @scale_AI org channels).
- What this paper is not.
  - Not a leaderboard claim. The number is the receipts that earn the read.
  - Not a critique of Pro. The bench-infrastructure note is one paragraph, structurally observed, not weaponized.

  - Not a contamination-clean science claim on Verified. Verified is contaminated for everyone; the loop port to Verified is a baseline, not a science claim.
- What this paper is.
  - A runnable agent harness that assembles a lineage of prior research (Peirce, Wald, Calcagno, Voyager, IRM) into one typed inquiry loop over a hypothesis-graph smem.
  - The engineering case study for the methodeutics textbook (Ch 15 in the textbook lineage).
  - An assembly contribution, falsifiable by execution: the harness runs end-to-end on industrial code under contamination discipline, or it doesn't.