# How Much Heavy Lifting Can an Agent Harness Do?: Measuring the LLM's Residual Role in a Planning Agent

Anonymous Author(s)

## Abstract

Agent harnesses—the stateful programs that wrap a language model and decide what it sees at each step—are now known to change end-to-end performance on a fixed model by as much as six times. That raises a question asked less often than it should be: how much of an agent's competence does the harness itself already carry, and how much genuinely still needs the LLM? We externalize a planning harness for noisy Collaborative Battleship into four progressively richer layers—posterior belief tracking, declarative planning, symbolic reflection, and an LLM-backed revision gate—under a common runtime, taking *win rate* as the primary metric and *F1* as secondary, and pre-specifying *heavy lifting* as the single largest positive marginal to the primary metric. Across 54 games, declarative planning carries the heavy lifting (+24.1pp win rate over a belief-only harness, zero LLM calls); symbolic reflection is mechanistically real but calibration-sensitive, with signed board-level effects up to ±0.140 F1 that cancel on aggregate; and LLM-backed revision activates on only 4.3% of turns with a bounded, non-monotonic effect. The contribution is methodological: once harness layers are made externally measurable, the LLM's role can be quantified as residual rather than assumed central.

***Keywords:*** agent harness, harness engineering, harness decomposition, structure-efficacy relationship, layer-wise ablation, LLM-based planning, residual LLM intervention, declarative runtime

## 1 Introduction

Harness engineering has become a load-bearing concept for LLM agents in 2025–2026. Recent industry and research reports show that the wrapper around a fixed language model—what gets stored, retrieved, and presented at each step—can change end-to-end performance by as much as six times on the same benchmark [3, 4, 13, 19]. A harness, in this sense, is a stateful program that wraps a language model and determines what context the model sees at each step. Production systems such as Claude Code [1], Gemini CLI, Codex, and OpenClaw lean heavily on harness structure—including shared procedural artifacts such as the `SKILL.md` spec [2]—to obtain reliable behavior from a general-purpose language model.

If the harness can change performance by six times, however, a question follows that is asked less often than it should be: *how much of an agent's competence does the harness itself already carry, and how much genuinely still needs the LLM?* Most current harnesses bundle belief tracking, action selection, reflection, and revision inside a single LLM-orchestrated loop [15, 17], which makes it hard to tell which layer is doing the heavy lifting.

Recent work on evolving agent systems—Agentic Context Engineering [20], Meta Context Engineering [18], Automated Design of Agentic Systems [8], and minimal-harness systems such as Terminus-KIRA [9] on Terminal-Bench [12]—*searches over* entire harnesses. It does not typically *decompose a single* harness into internally measurable layers, which is the gap this paper addresses.

We address this gap in one narrow, controlled setting. We take a planning harness and externalize it into four progressively richer layers: (L1) posterior belief tracking; (L2) declarative planning over the posterior through hypothetical transition evaluation and question timing; (L3) symbolic reflection via a confidence-gated, LLM-free revision mechanism; and (L4) LLM-backed revision activated only when the confidence gate opens. The planning domain is noisy Collaborative Battleship [6], a controlled mini-lab in which partial observability, belief update, budgeted questioning, and uncertainty-aware action selection coexist with turn-level consequences. Each layer is lifted into an inspectable declarative runtime so that its marginal contribution to the agent's performance, and the LLM's eventual invocation rate, can be observed rather than assumed.

***Pre-specified reporting criterion.*** Because layer contributions can diverge across metrics, we fix the reporting protocol before presenting results. We take *win rate* as the primary metric (it reflects whether the agent completes a game, the terminal success criterion of the domain) and *F1* as a secondary metric (it reflects local targeting precision conditional on a fixed shot budget). We pre-specify that a layer qualifies as *heavy-lifting* iff it contributes the single largest positive marginal to the primary metric among the measured layers; under this criterion at most one layer can be heavy-lifting. All per-layer comparisons in Section 4 are evaluated against this rule, and cases where F1 and win-rate ranks disagree are reported explicitly rather than resolved silently.

Across 54 games, the planning layer carries most of the heavy lifting under the above criterion (+24.1pp win rate

over a belief-only harness without a single LLM call); symbolic reflection is mechanistically real yet not net-positive on aggregate; and LLM-backed revision activates on only 4.3% of turns and yields a bounded, non-monotonic residual effect. The resulting picture is harness-first and LLM-residual. We make three contributions:

1. **Framework.** A four-layer decomposition protocol for a planning harness that isolates belief, planning, symbolic reflection, and LLM-backed revision as independently measurable runtime layers.
2. **Evidence.** Per-layer marginal contributions on 54 games of noisy Collaborative Battleship, with the LLM invocation rate emerging as a *dependent variable* of the confidence gate rather than a configured quota.
3. **Design bias.** A direct answer to "how much heavy lifting can an agent harness do?" in this setting: most of the agent's competence is carried by the first three layers of the harness, and the LLM's genuine responsibility is sparse, gated, and bounded.

## 2 What an Agent Harness Actually Does

***Harness as load-bearing substrate.*** A harness is a stateful program that wraps a language model and determines what context the model sees at each step [3, 13]. Because the harness can change benchmark performance on a fixed model by as much as six times [4], it is as load-bearing as model weights for many production use cases. The Agent Skills ecosystem [2] makes some procedural structure textual (`SKILL.md` is adopted by 30+ platforms), but most of the structural work in production harnesses is still load-bearing on the LLM inside the loop.

***What current harnesses bundle.*** Production planning harnesses routinely mix belief tracking, planning, reflection, and revision inside a single LLM-orchestrated loop. An LLM is asked, at each step, to decide what to believe, what to do, what went wrong, and how to repair. When the whole loop is driven by prompts, it is hard to diagnose which internal component of the harness carried—or broke—the agent. *This paper studies one narrow empirical question in this regime: how much heavy lifting can the harness itself do before the LLM is genuinely needed?*

***Decomposition vs. search.*** The recent harness literature splits along two axes. One axis is *search*: Agentic Context Engineering [20], Meta Context Engineering [18], Automated Design of Agentic Systems [8], and Terminus-KIRA [9] evolve or design entire harnesses, asking *which* macro-harness performs best on a benchmark such as Terminal-Bench [12]. The orthogonal axis, addressed here, is *decomposition*: holding a single harness fixed, lifting each internal layer into a separately addressable runtime object, and asking *what each layer contributes independent of the LLM*. Search and decomposition are complementary—search yields the right macro-structure; decomposition yields per-layer attribution—but they require different instrumentation. Search ranks harnesses by end-to-end pass rate; decomposition requires the harness to be lifted into objects whose preconditions and effects are externally observable. The unit of measurement, not the unit of optimization, is what changes.

**Table 1.** The four harness layers and the prior-work line each isolates. Decomposition is what makes each $\Delta_i$ measurable against a fixed substrate rather than against a different bundle.

| Layer | What it isolates | Closest prior-work analog |
|---|---|---|
| L1. Belief | Posterior over hidden world state | Bayesian experimental design; [6] belief backend |
| L2. Planning | World-model rollout for action and question scoring | LLM+P [11], WorldCoder [16] |
| L3. Symbolic reflection | Confidence-gated, LLM-free in-episode revision | Reflexion [15], MIDCA [5] |
| L4. LLM-backed revision | Conditional LM intervention on the same gate | ReAct [17] |

***Four layers, four prior-work analogs.*** The four layers we ablate are not invented from scratch; each isolates a line of prior work that current harnesses bundle into a single LLM-orchestrated loop (Table 1). L1 isolates the Bayesian belief layer of classical experimental design and the original Battleship harness [6]. L2 isolates the program-guided / world-model planning layer of LLM+P [11] and WorldCoder [16], with the LLM lifted out so the planning layer's marginal can be measured against a fixed belief backend. L3 isolates the meta-cognitive / self-reflective layer of Reflexion [15] and MIDCA [5], replacing the LM-emitted reflection with a confidence-gated symbolic mechanism so the gate's effect is measurable independent of LM output quality. L4 reattaches LM intervention as a *conditional residual* on the same gate—in the spirit of ReAct [17], but bounded to turns where the symbolic substrate has already declared itself uncertain. Each layer can therefore be ablated against its prior-work analog without changing the underlying mechanism.

***Why bundling obscures attribution.*** Let $\mathcal{H} = (B, P, R, V)$ denote a planning harness decomposed into belief, planning, reflection, and revision components, and let $\Phi(\mathcal{H})$ be end-to-end performance on a fixed metric. The per-layer marginal contribution of component $i$ is $\Delta_i := \Phi(\mathcal{H}) - \Phi(\mathcal{H}_{-i})$, where $\mathcal{H}_{-i}$ is the harness with component $i$ removed or replaced by a neutral substitute. Current harnesses fall into one of two regimes:

- *Bundled.* A single LLM call $\ell(\cdot)$ jointly emits belief updates, action proposals, reflective judgments, and revisions, so

$$\Phi(\mathcal{H}_{\text{bundle}}) = \Phi\big(\ell(s_t;\, \pi)\big),$$

where $\pi$ is a prompt pattern. The individual $\Delta_i$ are not identifiable from end-to-end traces alone, because removing any one role from the prompt also changes the others through their shared generation distribution.

- *Layered.* Each component is a separately callable runtime object with its own inputs, outputs, and (optionally) an LLM dependency, so

$$\Phi(\mathcal{H}_{\text{layered}}) = \Phi\big(B(s_t),\ P(\cdot),\ R(\cdot),\ V_{\text{det}}(\cdot) \oplus V_M(\cdot)\big|_{\text{gate}}\big),$$

and each $\Delta_i$ is directly measurable by a single-component ablation. Attribution is well-posed iff each component is lifted out of the shared LLM context.

This paper operates in the layered regime and measures $\Delta_i$ for $i \in \{B, P, R, V\}$ on a shared Battleship backend.

## 3 Externalizing the Planning Harness

***Preliminaries: state, signals, and gates.*** Let $s_t$ denote the game state at turn $t$ and $o_t$ the observation (a noisy hit/miss return or a question answer). The harness maintains a particle-approximated posterior $B_t$ over hidden ship placements (500 particles, Metropolis–Hastings backend), a per-turn predictive error $e_t^{\text{pred}}$ and calibration error $e_t^{\text{cal}}$, and their EMAs $\bar{e}_t^{\text{pred}}, \bar{e}_t^{\text{cal}}$ with coefficient $\alpha \in (0, 1]$. We define the runtime-computed *model confidence*

$$c_t \ := \ 1 - \tfrac{1}{2}\big(\bar{e}_t^{\text{pred}} + \bar{e}_t^{\text{cal}}\big) \ \in \ [0, 1],$$

and say the *revision gate is open* at turn $t$ iff (i) $c_t < \tau$ for the current threshold $\tau$, (ii) the low-confidence streak has length $\geq k_{\text{streak}}$, (iii) the revision cooldown counter is zero, and (iv) a `sim.next` counterfactual preview yields $\Delta\Phi \geq \delta_{\min}$ against the current policy. All four conditions are runtime-computable; none require an LLM. The layers differ only in (a) whether the gate's truth value is consulted and (b) who writes the revision patch when the gate opens (a symbolic preset vs. an LLM).

***Battleship as a decomposition lab.*** We study this in noisy Collaborative Battleship [6], where the agent maintains a posterior over hidden ship placements, chooses shots and information-gathering questions under a budget, and acts under noisy observations. Belief update, budgeted information gathering, uncertainty-aware action selection, and in-episode revision all appear at turn-level granularity in a compact domain. Battleship is not a canonical harness benchmark, and we do not use it as one. It serves here as a controlled mini-lab: a domain in which one harness layer can be varied at a time with interpretable turn-level consequences, ahead of replicating the decomposition in messier agent settings.

***Declarative runtime as instrumentation.*** The four layers in Table 2 are lifted into a declarative runtime whose primitives are *state*, *computed properties*, *guarded actions*, and *patches*. State is a typed record updated only by patches.

**Table 2.** Progressive externalization of a planning harness. Each row adds one layer that the harness carries without calling the LLM; the LLM enters only at L4, and its invocation rate is measured rather than pre-budgeted.

| Harness layer | Role added on top of the previous layer | LLM? |
|---|---|---|
| L1. Belief-only | Posterior belief tracking | No |
| L2. + Planning | Hypothetical transition evaluation and question timing | No |
| L3. + Symbolic reflection | Confidence tracking and guarded revision actions | No |
| L4. + LLM-backed revision | Residual revision when the confidence gate opens | Conditional |

Computed properties (e.g., `modelConfidence`, `sustained`, `positivePreview` in Listing 1) are pure functions over the state record, recomputed on every read and unable to have side effects. Actions declare their preconditions through an `available when` clause and their effects as patches; an action is legal at turn $t$ iff its precondition holds in the current state. The DSL is non-Turing-complete by construction—computed properties are total functions, actions are first-order with bounded patches, and there is no general loop or recursion. This restriction is what makes the layer ablation well-posed: toggling `revisionEnabled` cannot accidentally alter belief update or planning logic, because each layer's actions and computed properties are syntactically scoped to its own object. Hypothetical-transition evaluation is exposed as a `sim.next` primitive that scores a candidate action by its expected one-step posterior collapse rather than by a prompt-returned claim. The runtime itself is not the contribution of this paper; it is the substrate that makes per-layer marginals measurable on a shared backend, rather than confounded by a shared prompt context. Figure 1 shows the main loop; each phase corresponds to an externally recordable set of runtime events.

***Four harnesses.*** All four harnesses share the 500-particle Metropolis–Hastings posterior backend; they differ only in what is added on top:

- *Belief-only.* Fires at the highest-probability cell; asks no questions.
- *+ Planning.* Evaluates shoot and question actions via `sim.next` with a two-bucket question budget.
- *+ Symbolic reflection.* Adds an EMA-based confidence signal, a sustained-low-confidence gate, and a preset library of LLM-free revision actions (`coarse_roi_collapse`, `cluster_closeout_bias`, `late_diffuse_reprobe`, …).
- *+ LLM-backed revision.* Uses the same gate but may delegate the revision patch to a locally-served LLM; on protocol failure the runtime falls back to the symbolic preset.

Because belief-only and +Planning share the same belief backend, their contrast isolates the planning layer alone.

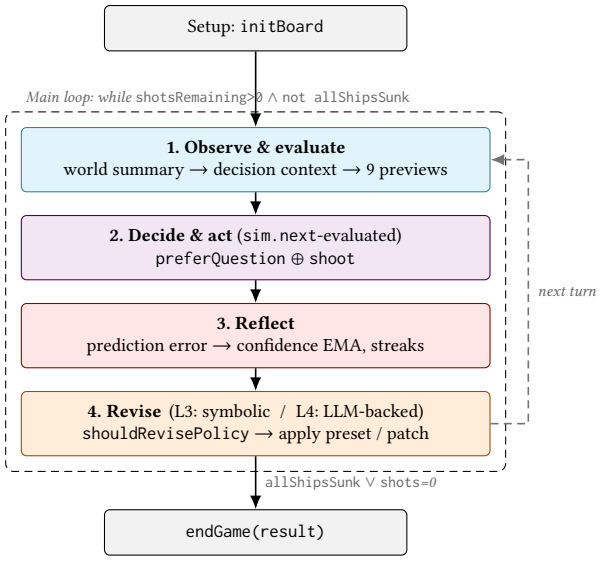


**Figure 1.** Planning harness main loop, externalized into declarative runtime events. Layers L1–L4 in Table 2 correspond to which phases are active: L1 uses only the belief-driven `shoot` in phase 2; L2 adds `sim.next`-based action selection with question timing; L3 adds the Reflect and Revise phases with symbolic presets; L4 replaces the Revise body with an LLM-backed patch when the confidence gate opens.

***Setup.*** All experiments use 8×8 boards with 14 ship cells, 40 shots, 15 questions, and $\varepsilon = 0.1$ observation noise. We report 18 boards × 3 seeds = 54 games. Reflective defaults are: EMA coefficient $\alpha = 0.25$, sustained low-confidence streak threshold 2, revision cooldown of 3 turns, minimum revision delta 0.01, and confidence threshold $\tau = 0.72$ by default (swept at $\{0.0, 1.0\}$ as endpoints). The LLM is locally served `gemma4:e4b` via Ollama. The board suite is a reimplementation, not the published one of [6]; the cross-suite comparison is isolated to §5 with the suite difference made explicit, rather than absorbed into the unified table.

## 4 How Far the Harness Gets Us

***Reading the unified table.*** Two facts carry the rest of this section. First, *non-reasoning LM-only Captains, regardless of model size up to a* 109*B MoE, fall below our no-LLM L1 baseline*: a ∼4B open-weights model lands at 0.263 F1 (below uniform Random; 6 report Random F1 0.317 on a similar 18-board suite), Llama-4-Scout at 0.353 (0 wins). Model size is not the missing ingredient—inference-time reasoning is. Second, *a small reasoning model (gpt-5-nano) closes most of the L1 gap but does not cross it* (29.6% WR), while *a mid-reasoning model (gpt-5-mini) reaches the L2 planning neighborhood* (77.8% WR vs. L2's 74.1%, with overlapping Wilson CIs). The same in-domain competence is therefore recoverable two ways: by adding planning structure with 0% LLM calls (L1→L2), or by spending an LLM call every turn at mid-reasoning class. Our decomposition isolates the no-LLM path and asks where the LLM still earns a residual *within* that path; the L4 row answers quantitatively—4.3% LLM rate, +0.005 F1 over L3—a small residual at $n$=54. The unified table reframes the question from "LLM vs. harness" to harness engineering: what does each declared layer buy, and how thinly is the LLM needed once those layers are in place?

**Table 3.** Unified within-suite comparison (54 games = 18 boards × 3 seeds; 8 × 8 noisy Battleship, $\varepsilon$=0.1; same MCMC belief backend). *LM-only Captains* call the LLM on every turn with no posterior, EIG, or decision rule; `openai/gpt-5-*` models use `reasoning_effort=medium` (default). *Harness layers* (this paper) progressively add structure on top of a shared belief; the LLM is gated to L4 only (`gemma4:e4b` via Ollama). Win-rate CI is Wilson 95% on the game-completion proportion. **Bold**: best per block.

| Captain / Harness | LLM Rate | Avg F1 | Win Rate | 95% CI | Avg Q |
|---|---|---|---|---|---|
| *LM-only Captains (no posterior / EIG / decision rule):* | | | | | |
| gemma3n:e4b (∼4B, no reasoning) | 100% | 0.263 | 0.0% | [0.0, 6.6] | 14.2 |
| Llama-4-Scout (109B MoE, no reasoning) | 100% | 0.353 | 0.0% | [0.0, 6.6] | 14.8 |
| gpt-5-nano (small reasoning) | 100% | 0.459 | 29.6% | [19.1, 42.8] | 12.4 |
| gpt-5-mini (mid reasoning) | 100% | **0.565** | **77.8%** | [65.1, 86.8] | 14.2 |
| *Harness decomposition (this paper):* | | | | | |
| L1: Belief-only | 0% | 0.522 | 50.0% | [37.1, 62.9] | 0.0 |
| L2: + Planning | 0% | 0.539 | **74.1%** | [61.1, 83.9] | 11.9 |
| L3: + Symbolic reflection (off) | 0% | 0.552 | 57.4% | [44.2, 69.7] | 8.0 |
| L3: + Symbolic reflection (on) | 0% | 0.551 | 55.6% | [42.4, 68.0] | 8.0 |
| L4: + LLM-backed revision ($\tau$=1.0) | 4.3% | **0.557** | 53.7% | [40.6, 66.3] | 8.9 |

***§4.1 The planning layer does the heavy lifting.*** *Key finding.* Adding declarative planning on top of a shared belief backend raises win rate by +24.1pp (50.0% → 74.1%) with zero LLM calls—the single largest layer effect in the decomposition, and the only contrast in which the Wilson 95% CIs of the two harnesses ([37.1, 62.9] vs. [61.1, 83.9]) are non-overlapping at $n$=54. Under the pre-specified criterion (§1), the planning layer is therefore the unique heavy-lifting layer in this decomposition.

*Mechanism.* The +24.1pp gap is not a generic "more compute" effect. L1 commits each turn to the cell of maximum posterior mass, which is the locally-greedy action when the posterior is sharp but offers no information value when the posterior is flat or multimodal. L2 changes the decision rule in two coupled ways. First, each candidate is scored through a `sim.next` preview that ranks actions by *expected one-step posterior collapse* rather than immediate hit probability. Second, the action set includes region-DSL questions drawn under a two-bucket budget so that questions are *timed*—early-game ROI narrowing and late-game cluster closeout—rather than uniformly rate-limited. The asymmetric gain pattern (+0.017 F1 vs. +24.1pp win rate) is what this mechanism predicts: L2 does not change local targeting precision much, but it changes how often the agent escapes the flat-posterior regime in which L1's greedy rule has no edge. The unique

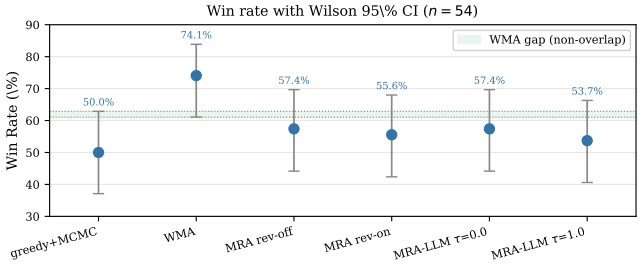


**Figure 2.** Win rate per harness layer with Wilson 95% binomial confidence interval ($n$=54; harness block of Table 3). The L1→L2 step is the only vertical jump in the harness; within the reflective family (L3, L4) the intervals are flat against each other and the LLM-backed L4 sits in the same band as the LLM-free L3 baselines. The picture is what "the harness does the heavy lifting, not the LLM" looks like at the level of confidence intervals.

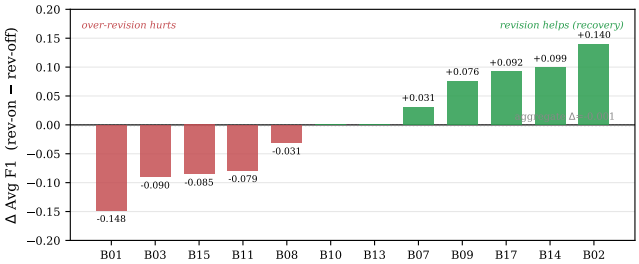


**Figure 3.** Per-board ΔF1 (symbolic reflection on minus off) across the 12 boards recoverable from the registry top/bottom-5 overlap, sorted. Green bars are recovery boards (revision helps); red are over-revision boards (revision hurts). The dotted grey line is the aggregate Δ=−0.001. The bimodal, signed-both-ways shape is visual evidence that the right diagnosis of the flat aggregate is calibration, not absence of mechanism.

non-overlapping CI in the decomposition is the shape of that effect at the suite level, rather than a small mean shift.

Figure 2 renders this concentration as a single picture. The L1→L2 step is the only vertical lift in the decomposition; once one moves into the reflective family the win-rate intervals are flat against each other, with the LLM-backed layer (L4) overlapping the LLM-free L3 baselines. Adding an LLM revision layer on top of an already-reflective harness produces parallel motion, not a second jump—and "the harness does the heavy lifting, not the LLM" is exactly the shape that picture has. The F1 gain at L1→L2 is correspondingly smaller (+0.017), producing a *last-mile asymmetry*: planning mostly converts borderline games into wins rather than changing local targeting precision. One might object that +Planning simply benefits from asking more questions. The objection misses what the planning layer does: each question is selected and timed by `sim.next` under declared budget constraints, not issued by a generic question policy.

A second, stronger objection conflates "harness-first" with "LLM-free" and reads the +24.1pp lift as "you just built a good solver." The unified table (Table 3) refines this within our own suite. *Non-reasoning* LM-only Captains, including a 109B MoE (Llama-4-Scout: 0.353 F1, 0 wins), all fall below L1 (0.522 F1, 50.0% WR); a *small reasoning* model (gpt-5-nano) closes most of the gap but does not cross it (29.6% WR); a *mid-reasoning* model (gpt-5-mini) reaches the L2 planning neighborhood at 100% LLM cost (77.8% WR, CI overlapping L2's). The harness's contribution is therefore not "you don't need an LLM"—it is more pointed: the same in-domain competence is recoverable at 0% LLM cost by adding ∼0.017 F1 / +24.1pp WR of explicit planning structure on top of a shared belief, which is the substance of the L1→L2 contrast. Our decomposition then asks where the LLM still earns a residual *within* that no-LLM path: at L4 the gate fires on 4.3% of turns and adds +0.005 F1 over L3, a small qualitative residual at $n$=54. Read together, the unified table reframes the question from "LLM vs. harness" to harness engineering: what does each declared layer buy, and how thinly is the LLM needed once those layers are in place?

***§4.2 Symbolic reflection is mechanistically real but calibration-sensitive.*** *Key finding.* Symbolic reflection produces large signed board-level effects (up to ±0.140 F1) that cancel on aggregate (−0.001 F1, −1.8pp win rate). The cancellation is itself a diagnosable calibration finding—two regimes firing in opposite directions—rather than absence of mechanism.

What is declared is a runtime legality condition (Listing 1), not a prompt heuristic: revision actions are only available when low confidence is both sustained and beaten by a concrete counterfactual preview. Board-level effects are therefore large and signed in both directions:

| **Recovery** ΔF1 | | **Over-revision** ΔF1 | |
|---|---|---|---|
| B02 | +0.140 | B01 | −0.148 |
| B14 | +0.099 | B15 | −0.085 |
| B17 | +0.092 | B11 | −0.079 |
| B09 | +0.076 | | |

Figure 3 plots these signed effects across all 12 recoverable boards. The distribution is sharply bimodal—large positive bars on recovery boards, large negative bars on over-revision boards, with the aggregate Δ= − 0.001 landing exactly between the two regimes. A mechanism that did nothing would produce a flat band at zero; the actual picture is two clearly separated regimes, in opposite directions, that happen to cancel on average. The reflection mechanism is firing on both kinds of board; what is missing is a trigger calibrated to fire only on the recovery kind. The B17-seed0 trace (Appendix B) makes the mechanism concrete. With symbolic revision enabled, `coarse_roi_collapse` fires at turn 2 and `cluster_closeout_bias` fires at turns 12 and 15, yielding

**Listing 1.** Symbolic reflection as declared runtime legality (defaults: $\tau = 0.72$, streak $\geq 2$, cooldown = 3, $\delta_{\min} = 0.01$). Confidence, sustained low-confidence, and positive preview are all computed properties rather than prompt-returned claims; `applyRevision` is a guarded action, not an LM call.

```
computed modelConfidence = 1 - (predictionErrorEMA
                              + calibrationErrorEMA) / 2
computed confident       = modelConfidence >= confidenceThreshold
computed sustained       = lowConfidenceStreak >= 2
computed canRevise       = not confident
                           and (cooldownRemaining == 0)
computed revisionRequested = canRevise and sustained
                             and positivePreview
                             and (revisionKind != "")
computed shouldRevise    = revisionEnabled and revisionRequested

action applyRevision available when shouldRevise:
    patch policyParameters <- nextParameters
    patch cooldown         <- cooldownTurns
```

a win at F1 0.609. With revision disabled, no revision action fires and the agent exhausts all 40 shots for F1 0.333. Declaring reflection as a runtime layer rather than a latent prompt pattern is what makes board-level diagnosis like this possible: the mechanism exists; what is missing is a calibrated preset library.

***Beyond a single board.*** The B17 trace is not an isolated anecdote. Across the four recovery boards (B02, B14, B17, B09) the revision actions that fire are either `coarse_roi_collapse` at an early turn ($t \leq 4$) or `cluster_closeout_bias` after a cluster of mid-game hits—i.e., situations where the posterior genuinely spends several consecutive turns in a flat or miscalibrated regime. In the three over-revision boards (B01, B15, B11), revision instead activates on a transient low-confidence dip that resolves without intervention, diverting shots from an already-correct local targeting phase. The preset library is therefore well-calibrated for *sustained* ambiguity but miscalibrated for *transient* confidence dips, which is consistent with the EMA-based $c_t$ responding faster than the cooldown and streak guards can filter. This is a calibration failure of the revision trigger, not a structural failure of the reflection mechanism.

***§4.3 The LLM is a sparse, non-monotonic residual.*** *Key finding.* The LLM is invoked on only 4.3% of turns and its marginal effect is bounded and non-monotonic—sufficient evidence, in this setting, that the LLM is not the centre of gravity of the harness.

The confidence gate controls everything. At $\tau = 0.0$ it never opens (the system collapses onto symbolic revision-off); at $\tau = 1.0$ it opens on a measured 4.3% of turns. The tradeoff at $\tau = 1.0$ is itself interesting: F1 is the highest of the reflective family (0.557) while win rate is the lowest (53.7%). LLM revisions sharpen local targeting at the cost of game-level completion. An 18-game vs. 54-game comparison (Appendix A) preserves the qualitative pattern (non-monotonic, bounded) but shows that the exact tradeoff is unstable under small $n$: at 18 games, $\tau = 1.0$ achieves both the highest F1 *and* the highest win rate; at 54 games, the win-rate advantage reverses while the F1 ranking is preserved.

***§4.4 What harness decomposition buys.*** Not every added layer is net-positive in aggregate—and that is the point. What decomposition buys is that mixed outcomes become *interpretable* rather than averaged into an opaque pass rate:

- Symbolic reflection: aggregate-neutral yet decisive on specific boards.
- LLM-backed revision: F1-helpful yet win-rate-harmful.
- Hidden L2→L3 gap: the raw win rate also drops from 74.1% (+Planning) to 57.4% (−16.7pp) when the reflection layer is installed even with its gate off. As the Limitations section notes, this transition changes the question-budget policy as a confound, so we do *not* read this as a pure reflection-layer cost—but decomposition is what surfaces it as a question to answer rather than a pattern to hide.

These patterns would not be observable in a single end-to-end pass rate. In summary, when harness layers are externalized, failure turns into diagnosable signal rather than opaque pass/fail: the primary-metric heavy-lifting layer is identified (L2), the board-level direction of L3's signed effect is made visible, L4's residual role is quantified as a dependent 4.3% invocation rate, and confound-induced gaps (L2→L3) are exposed rather than absorbed into an aggregate number.

## 5 Discussion

***Harness-first, LLM-residual as a design bias.*** The decomposition supports a concrete design bias for LM-assisted planning in settings where belief, action evaluation, and revision admit symbolic representation: exhaust declarative structure first, and reserve the LLM for gated residual revision. This sharpens the direction of program-guided planners [11, 16]: in a well-specified planning domain, the residual role of the LLM can be significantly smaller than end-to-end agent architectures suggest.

***Measurement, not optimization, is the contribution.*** Once the harness is externalized, prediction error, model confidence, revision eligibility, and revision outcomes become first-class inspectable variables rather than latent prompt effects. Symbolic reflection is the clearest illustration: it is not yet net-positive on aggregate, yet decomposition reveals where it helps, where it hurts, and why calibration matters—as the B17-seed0 trace makes concrete. Aggregate-neutral and uninformative are different things, and the difference is only visible when the layer is lifted out.

***Relation to metacognitive architectures.*** The decomposition connects to a long line of metacognitive systems—Soar [10] integrates planning with impasse-driven meta-level

intervention, MIDCA [5] separates cognitive and metacognitive loops, HYDRA [14] detects environment novelty and repairs PDDL+ domains across episodes, and self-aware agents [7] track world-model prediction error to guide exploration. Three differences matter for our setting: (i) the metacognitive loop is declared *inside* the harness as guarded actions and computed signals rather than delegated to an LM wrapper or a separate cognitive cycle; (ii) revision is in-episode rather than across-episode; and (iii) the reflective loop operates without LLM calls by default, with LLM intervention attached only as a conditional effect of the same gate. This is what makes layer-by-layer ablation possible without changing the underlying mechanism.

***F1/win-rate divergence as a budget tension.*** The F1 and win-rate signals diverge across the reflective family in a way that has a planning explanation. +Planning achieves its high win rate while asking about 11.9 questions per game; both symbolic and LLM-backed reflective variants ask about 8, with the LLM-backed variant recovering only to 8.9 at $\tau$=1.0. Question budget is what closes out borderline games, so a revision—whether symbolic or LLM-backed—competes with question allocation for a finite per-game turn supply. LLM-chosen revisions in particular sharpen local targeting (highest F1 in the reflective family at 0.557) but sometimes spend turns the symbolic question budget would have spent differently to finish the game. The decomposition makes this tradeoff visible at the layer level rather than absorbing it into a single end-to-end pass rate.

***Relation to the original Battleship benchmark.*** A substantial F1 gap remains relative to the strongest published configuration of [6] on their suite: Llama-4-Scout combined with their full Bayesian harness (Bayes-Q + Bayes-M + Bayes-D) reaches F1 0.764. The plausible contributor we cannot rule out is the absence of language-informed belief construction (their LLM-emitted Python-program question pool produces richer EIG candidates than our region-based DSL)—an orthogonal LLM-in-planning axis that our decomposition does not attempt to cover. The decomposition isolates the residual role of the LLM *within the planning loop*; it does not speak to the role of the LLM *within the belief-construction step*, which remains an open and complementary question.

## 6 Limitations

Five caveats bound the scope of these claims.

- **Single domain.** All results are from Battleship; the protocol is untested elsewhere. Battleship is also a probabilistic-search problem in which MCMC posteriors and `sim.next`-evaluated planning are a particularly natural fit, so the planning layer's +24.1pp lift may overstate what declarative planning recovers in less search-friendly domains. The same MCMC-friendly properties also stress-test the LLM's residual role from the opposite side: the unified table (Table 3) shows LM-only Captains span F1 0.263–0.565 across non-reasoning and reasoning model classes within our suite, so the small L4 residual we measure is consistent with the symbolic substrate doing most of the planning work *in this domain*, not with the LLM being unhelpful in general.
- **Sample size.** 54 games expose qualitative patterns but do not support tight confidence intervals; the 18-game vs. 54-game divergence at $\tau$=1.0 (Appendix A) is a concrete reminder of small-$n$ instability.
- **Preset calibration.** The current symbolic library is a first calibration pass rather than a final policy, which is why symbolic reflection is not net-positive on aggregate. Per-board signed effects suggest the mechanism is real and the gap is in calibration, not architecture.
- **LLM choice.** The residual was measured with one locally served 9B model (`gemma4:e4b`); the bounded, non-monotonic tradeoff may shift under larger cloud-served LLMs, and the threshold sweep is sparse rather than fully characterized.
- **Question-policy confound.** The transition from +Planning to +Symbolic reflection also changes the question-budget policy (two buckets to three plus exploit threshold), so the reflection ablation isolates "reflection given its own question policy" rather than reflection over a shared question policy. A cleaner isolation, in which the question policy is held fixed across the L2→L3 transition, is left for future work.

## 7 Conclusion

Once a planning harness is externalized into measurable layers—belief tracking, declarative planning, symbolic reflection, and LLM-backed revision—the question "how much heavy lifting can the harness do?" becomes a layer-level decomposition rather than an opaque pass/fail. In this setting, the planning layer carries the largest single effect (+24.1pp win rate over a belief-only baseline, with zero LLM calls); symbolic reflection is mechanistically real but calibration-sensitive at the current preset library; and LLM-backed revision is a sparse, gated residual whose marginal effect is bounded and non-monotonic. The corresponding design bias is direct: declare what you can, reflect symbolically where possible, and reserve the LLM for the residual that the declared substrate cannot resolve. In that design, the load-bearing question for an LLM agent shifts from "does the agent use an LLM?" to *where* LLM intervention is empirically justified, and a declared harness is the instrumentation that makes the answer measurable. Future work will (1) replicate the decomposition in a second harness domain with different action and observation geometry, (2) calibrate the symbolic

preset library so reflection becomes net-positive on aggregate, and (3) characterize the full $\tau$-benefit curve across LLM sizes and quantizations.

# A Threshold Sweep Detail

**Table 4.** Threshold sweep across evaluation scales (§4.3). Thresholds 0.0 and 0.5 collapse into a no-LLM basin; threshold 1.0 activates LLM revision on a measured 4–5% of turns. The 18-game and 54-game rows diverge at $\tau$=1.0, underscoring the need for larger-scale evaluation while preserving the qualitative finding of non-monotonicity.

| Scope | $\tau$ | LLM Rate | Avg F1 | Win Rate |
|---|---|---|---|---|
| 18 games | 0.0 | 0.0% | 0.522 | 50.0% |
| 18 games | 0.5 | 0.0% | 0.522 | 50.0% |
| 18 games | 0.72 | 1.1% | 0.515 | 50.0% |
| 18 games | 1.0 | 4.6% | 0.569 | 61.1% |
| 54 games | 0.0 | 0.0% | 0.552 | 57.4% |
| 54 games | 1.0 | 4.3% | 0.557 | 53.7% |

# B B17-seed0 Trace Summary

**Symbolic reflection on (win, F1** 0.609**).** Turn 2: `coarse_roi_collapse`. Turns 12 and 15: `cluster_closeout_bias`. **Symbolic reflection off (loss, F1** 0.333**).** No revision action fires; the agent exhausts 40 shots without finishing.

**Figure 4.** Qualitative event trace for the B17-seed0 case discussed in §4.2. The trace is intentionally compact because the full runtime log is best inspected as an artifact bundle.

# References


[1] Anthropic. 2025. Claude Code: An agentic coding tool. https://www.anthropic.com/claude-code.

[2] Anthropic and community contributors. 2025. agentskills/agentskills. GitHub repository https://github.com/agentskills/agentskills. Specification and documentation for Agent Skills.

[3] Birgitta Böckeler. 2026. Harness Engineering. https://martinfowler.com/articles/exploring-gen-ai/harness-engineering.html. martinfowler.com.

[4] Can Bölük. 2026. I Improved 15 LLMs at Coding in One Afternoon. Only the Harness Changed. https://blog.can.ac/2026/02/12/the-harness-problem/.

[5] Michael T. Cox, Dana Dannenhauer, Donald Perlis, Stuart C. Shapiro, and Murugesan Sundaram. 2016. MIDCA: A Metacognitive Integrated Dual-Cycle Architecture. In *AAAI Workshop on Metacognitive Machine Learning*.

[6] Gabriel Grand et al. 2025. Collaborative Battleship with LLMs as Bayesian Experimental Designers. Preprint.

[7] Nick Haber, Damian Mrowca, Stephanie Wang, Li Fei-Fei, and Daniel L. K. Yamins. 2018. Learning to Play with Intrinsically-Motivated, Self-Aware Agents. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[8] Shengran Hu, Cong Lu, and Jeff Clune. 2025. Automated Design of Agentic Systems. In *International Conference on Learning Representations (ICLR)*.

[9] KRAFTON AI and Ludo Robotics. 2026. Terminus-KIRA: Boosting Frontier Model Performance on Terminal-Bench with Minimal Harness. https://github.com/krafton-ai/kira

[10] John Laird, Allen Newell, and Paul Rosenbloom. 1990. Soar: An Architecture for General Intelligence. *Artificial Intelligence* 33, 1 (1990), 1–64.

[11] Bo Liu et al. 2023. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. Preprint.

[12] Mike A. Merrill et al. 2026. Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces. *arXiv preprint arXiv:2601.11868* (2026).

[13] OpenAI. 2026. Harness engineering: leveraging Codex in an agent-first world. https://openai.com/index/harness-engineering/. OpenAI Blog.

[14] Wojciech Piotrowski et al. 2023. HYDRA: Adaptive Operation in Evolving Open Worlds. Preprint.

[15] Noah Shinn, Federico Cassano, Emanuel Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[16] Haotian Tang, David Key, and Kevin Ellis. 2024. WorldCoder: A Model-Based LLM Agent. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[17] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*.

[18] Haoran Ye, Xuning He, Vincent Arak, Haonan Dong, and Guojie Song. 2026. Meta Context Engineering via Agentic Skill Evolution. *arXiv preprint arXiv:2601.21557* (2026).

[19] Justin Young. 2025. Effective harnesses for long-running agents. https://anthropic.com/engineering/effective-harnesses-for-long-running-agents. Anthropic Engineering Blog.

[20] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, V. Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and K. Olukotun. 2025. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. In *arXiv preprint arXiv:2510.04618*.