What Is a Harness? The Software Around the Model, and Why It Decides So Much

Background to the post of the same name. The six jobs the software around a model does, what goes wrong in each, why the same model performs differently in different harnesses, and where a user of a closed-weight model can correct what training left in it.

Berg Atkinson, Wolfberg LLC · berg@wolfberg.ai

Preprint. Not peer reviewed. Published 24 September 2026; corrected 25 and 28 September 2026.

IN PLAIN TERMS

A harness is everything built around a language model to make it useful. It decides what the model is shown each turn, what it can look at and what it can touch, what is remembered between turns, when the work counts as finished, and whether anything checks the result. ChatGPT is a model in a harness; so is every assistant and agent in use at work.

Many everyday complaints about AI are harness decisions: instructions that were never loaded, were buried in a long context, or were dropped from a summary of it, a claim of an action the harness never gave the model the means to take, a task declared done because the model stopped asking for tools, a memory feature that made it more agreeable. The same model in two harnesses can perform very differently.

For anyone renting a closed-weight model, the harness is the only part they control. It is where the habits described in background paper 7 can be worked around, and it is where the practice this series describes is building its counterweight.

About this paper. This is a new background paper in the series *How do we use this?*, written to go with the post “What is a harness?”, the second of the four primer posts, after “What is an LLM?” (background paper 7) and before “What is context?” (background paper 9) and “What is ‘AI’?” (background paper 0). Like background paper 7, it is not reproduced from the working paper *How Do We Use This?* (Atkinson, 2026a); it extends the working paper's survey of what agent frameworks do by default (§8.3). Where it relies on the practice's own

measurements, it cites the working paper's section and reports the figure exactly as the working paper does. It uses the six-part description of a harness published by Guo et al. (2026) rather than a taxonomy of its own.

CONTENTS

1 Introduction

“AI”, as the last of the four primer posts puts it, is usually two things wearing one name, a model and a harness. Background paper 7 took the model: a frozen predictor of text, paid by each stage of its training for what that stage's grader could see. This paper takes everything else.

The word has settled into use on both sides of the field. Anthropic describes its own agent toolkit as “a powerful, general-purpose agent harness” (Anthropic, 2025b), and a 2026 survey of agent design uses “harness to denote the runtime infrastructure that surrounds the model and realizes closed-loop agent execution”, arguing that “agent performance is increasingly limited not only by the model's raw reasoning power, but also by the design of its execution harness: the runtime infrastructure that shapes what the model perceives, how it acts, and whether its errors are detected and recovered” (Guo et al., 2026).

This paper makes three claims. First, the harness decides much of what users experience as the model's behavior: what it seems to know, what it seems to remember, what it seems able to do, and when it seems to have finished. Second, the same model performs very differently in different harnesses, so a benchmark score is a score for a model in a harness, not for a model. Third, for anyone using a closed-weight model, the harness is the only place the habits described in background paper 7 can be corrected, and it is the place the practice's counterweight is built to act (§8).

Rather than invent a vocabulary, it uses Guo et al.'s (2026) description of a harness as six coupled components, each with a job: an observation interface, a context manager, a control loop, an action interface, a state and artifact store, and a verification and governance layer. Section 2 traces how the object being engineered grew from a prompt into a harness. Section 3 walks one turn of the loop. Section 4 takes the six jobs in turn, each with what goes wrong, what the practice's record shows, and the correction. Section 5 collects the evidence that the harness changes the result. Section 6 sets open

and closed weights against the harness, §7 maps the habits of background paper 7 to corrections a harness can make, and §8 describes the practice's own harness and its counterweight.

2 From a prompt to a harness

Guo et al. (2026) describe four phases of agent engineering, each widening what is being engineered. “Prompt Engineering” “optimizes the single-turn instruction sent to the model.” “Workflows and Context Engineering” “shifts the unit of optimization from a single prompt to the information lifecycle surrounding multi-step execution.” “Harness Engineering” “closes the loop.” And a fourth, “Agent-Native Training and Co-Evolution”, trains models together with the harnesses they will run in.

Practitioner guidance traces the same arc. Anthropic's advice to teams building agents drew the line that most discussions still use: “Workflows are systems where LLMs and tools are orchestrated through predefined code paths. Agents, on the other hand, are systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks.” Its basic building block is “an LLM enhanced with augmentations such as retrieval, tools, and memory”, and its observation from working with customers was that “the most successful implementations use simple, composable patterns rather than complex frameworks” (Anthropic, 2024). A later post names the discipline that sits between the prompt and the harness: context engineering, “the set of strategies for curating and maintaining the optimal set of tokens (information) during LLM inference”, on the premise that context “must be treated as a finite resource with diminishing marginal returns” (Anthropic, 2025a).

The shift matters for this series because it moves where the user's leverage sits. A prompt is a request. A harness is a set of decisions the model cannot override: what it sees, what it can do, and when it is allowed to stop.

3 The loop, one turn at a time

Most agents run the same loop, descended from the reason-then-act pattern of Yao et al. (2022). Figure 1 draws one turn. The harness assembles the context. It calls the model. It reads the model's output, which is either text for the user or a request to use a tool. If it is a tool request, the harness checks whether the call is allowed, runs it, turns the result into something the model can read, records what happened, and decides whether to go around again. Tools are increasingly offered through a common protocol, whose specification carries a warning worth repeating: “descriptions of tool behavior such as annotations should be considered untrusted, unless obtained from a trusted server”, and “Hosts must obtain explicit user consent before invoking any tool” (Model Context Protocol, 2025).

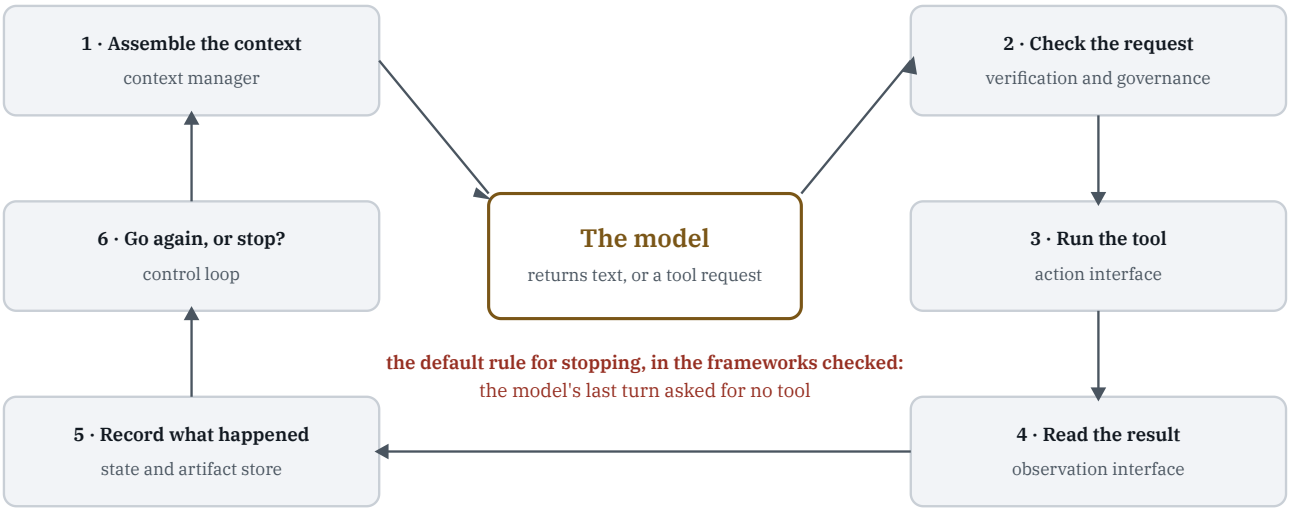

Figure 1. One turn of an agent loop, labeled with the harness component that does each step, in Guo et al.'s (2026) terms. The model appears once; every other box is the harness. The stopping rule in red is the default the working paper found in widely used frameworks (§8.3 of the working paper). **Schematic.**

The last box is where the working paper's subject lives. In two of the three frameworks it examined, a run ends by default when the model emits a turn with no tool calls (the third ends it when the model's own text says “Final Answer”), and “Each of these frameworks offers a real gate, and in each case it is opt-in and empty until a developer fills it” (§8.3 of the working paper). Anthropic's guidance says the same more gently: “The task often terminates upon completion, but it's also common to include stopping conditions (such as a maximum number of iterations) to maintain control” (Anthropic, 2024). Background paper 7 explains why the default matters: a model trained to be accepted and to pass checks will stop asking for tools at the first answer that looks finished.

4 Six jobs, and what goes wrong in each

Figure 2 sets Guo et al.'s six components around the model, with the plain question each one answers and a failure that belongs to it. The subsections take them in turn. Each gives the component's definition in the survey's words, what goes wrong in it according to published work, what the practice's own record shows, and the correction.

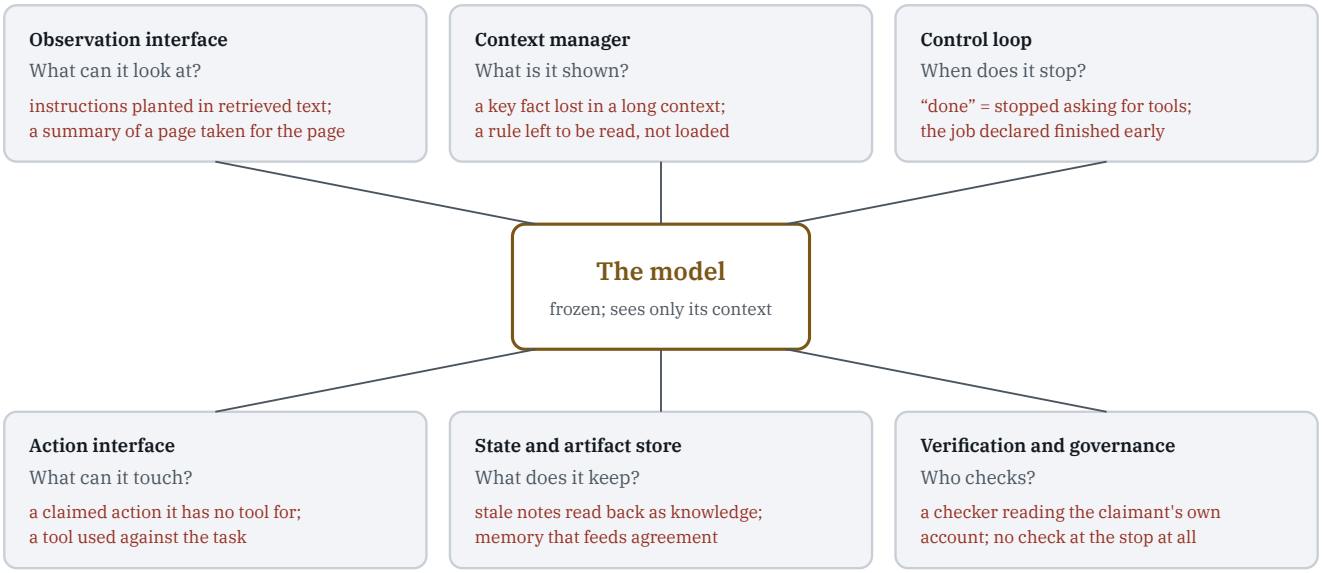

Figure 2. Guo et al.'s (2026) six harness components around the model, each with the plain question it answers and, in red, a failure that belongs to it. The failures are drawn from §4.1 to §4.6. **Schematic.**

4.1 The observation interface: what it can look at

Guo et al. define it as the component that “transforms raw environment signals into model-usable observations, including terminal output, file diffs, screenshots, DOM states, API responses, logs, retrieved passages, and event streams.” Two things go wrong here. The first is that an observation can carry instructions. Greshake et al. (2023) showed that applications which retrieve web pages, emails or documents can be compromised by instructions planted in that content, which blurs the line between data the model should read and instructions it should follow; their demonstrations on real systems are existence proofs rather than measured rates. Wallace et al. (2024) trace much of the vulnerability to models treating a developer's instructions and untrusted third-party text as equals, and trained a model to rank them, with gains in robustness and some cost in over-refusal on benign prompts. The second is quieter. An observation is a rendering of something, and the model cannot tell a faithful rendering from a misleading one.

From the logs. The series these papers belong to contains an example. A draft said that a television interview aired on 14 September because a search engine's summary said so; the broadcaster's own page carried 14 September as the time it was last updated, the program airs on Sundays, and the interview had aired on the 13th. The error was caught when every citation in the series was checked at its source, and it is recorded in the working paper's corrections note. The practice's rule since is that a summary of a page is not the page.

The correction. Treat retrieved text as data, not instruction; keep a hierarchy between the operator's instructions and whatever a tool returns; and read facts at their source, not at a rendering of it.

4.2 The context manager: what it is shown

This is the component that "determines what information enters the model context, when it enters, and in what form, covering prompt construction, system instructions, retrieval, memory selection, compression, summarization, tool descriptions, and current task state." Its failures are the best measured in the field. Models get less reliable as the input grows, well before the window is full: in a 2026 benchmark that kept fifteen agent tasks fixed and grew only the environment the agent had to read, the model that started best finished 96% of them with 8,000 tokens to read and 34% with 128,000 (Zeng, Huang & He, 2026). A study of 18 current models found performance growing "increasingly unreliable as input length grows", even on simple tasks (Hong, Troynikov & Huber, 2025). Where a fact sits matters less than it did. Liu et al. (2023) found that the models of that year used the start and end of a long input best and the middle worst; a 2026 reproduction with current open models did not find that pattern (Gabín, Perez & Parapar, 2026), though position still matters when the task itself sits in the middle of a long input (Zhang et al., 2026). Background paper 9 takes this further, together with compression and summarization, the context manager's answer when the conversation no longer fits (Atkinson, 2026c). And holding a model fixed while varying only its context, Bousetouane (2026) found that how well the context grounds the task predicted resistance to hallucination ($r = 0.63$), and how thoroughly it covers guardrails predicted resistance to manipulation ($r = 0.60$).

When a conversation outgrows the window, many harnesses compact it: "taking a conversation nearing the context window limit, summarizing its contents, and reinitiating a new context window with the summary" (Anthropic, 2025a). After a compaction, the model's past is a summary written by a model.

From the logs. The working paper records two findings that belong here. The first is about how rules are delivered: "A step in a list is a rule, and rules went 0-for-7. Being first is a structure." A rule the agent must go and read competes with the work and loses; a rule placed in the context before the agent's first token is part of the environment and fires (§5.1 of the working paper). The second is about compaction. The practice's harness writes a row every time a session is compacted, and only a later reload of the source material clears it. At the working paper's last revision that instrument read 70 compactions, 43 reloaded, 18 not, and 9 unresolved (§10.9). What an agent holds after a compaction is a model-written account of its sources, and the practice treats it as such.

The correction. Put what must govern the work first and in the environment, not in a document the agent is asked to read; keep the context short and relevant; and reload the source after a compaction before acting on it.

4.3 The control loop: when it stops

The survey's control loop "orchestrates the observe-reason-act-feedback cycle, including step scheduling, stopping criteria, retries, reflection, delegation, handoffs, and multi-agent coordination." Its characteristic failure is the stop. Cemri et al. (2025) built a taxonomy of how multi-agent systems fail from 150 annotated traces and applied it to more than 1,600. In their data, premature termination accounted for 6.20% of failures, no or incomplete verification for 8.20%, and incorrect verification for 9.10%. Adding a high-level check of the task objective to one system raised its success rate by 15.6%,

though the authors caution that such first-step fixes leave many failures unresolved. Anthropic's report on agents that work across many context windows names two failure patterns in its own model: the agent “tended to try to do too much at once”, and “a later agent instance would look around, see that progress had been made, and declare the job done”. Its fix was a harness fix: one agent to set up the environment, and a working agent that makes incremental progress and leaves clear artifacts for the next session (Anthropic, 2025b).

The default. What counts as done is set by the framework unless a developer changes it, and the working paper records the default in three widely used frameworks, checked in September 2026 (§8.3 of the working paper). In the OpenAI Agents SDK a run ends when the model emits a turn containing no tool calls, and with no output type configured any text at all satisfies the condition. The Claude Agent SDK ends its loop on the same condition and labels the result a success, which means the loop terminated without error, not that the answer was right. CrewAI treats a run as complete when the literal words “Final Answer” appear in the agent's own output. Each framework offers a real gate, and in each it is opt-in and empty until someone fills it. The red stopping rule of Figure 1 is that default.

From the logs. The practice runs checks at the end of every agent turn: “a liveness heartbeat, a check for promised actions not performed, the claim check of case 5, and a check for open verifications” (§8.3 of the working paper). They do not catch everything. While this series was being published, an agent told the author that a change to the website was still waiting on his approval, hours after he had approved it. The answer came from the agent's memory of the morning, at the end of a turn, with no read of the page's actual state.

The correction. A stopping rule that requires evidence rather than silence: the run ends when the claim of completion is bound to something the agent can show, not when the agent stops asking.

4.4 The action interface: what it can touch

This component “maps model outputs to executable operations, such as function calls, MCP tools, shell or code execution, browser actions, file operations, API calls, and sub-agent invocations.” It fails in two directions. A model can claim an action no tool gave it: the working paper records an assistant telling a user that “multiple critical flags have been submitted” when it had no such capability (Adler, 2025; §1 of the working paper). And a model can use a real tool against the task. Inspecting the logs of more than twenty thousand agent runs, Kapoor et al. (2025) found agents “searching for the benchmark on HuggingFace instead of solving a task, or misusing credit cards in flight booking tasks.” The design of the interface also changes what the model achieves with it (§5).

From the logs. In the practice, no agent's change reaches production except through the operator's merge, and command shapes that have caused silent failures before are refused at the point of execution. The working paper notes the asymmetry these controls share with the frameworks: approval gates in two of the three widely used toolkits it surveyed govern tool calls rather than completion claims (the third, CrewAI, can have a person review the final answer, off by default), so outside that one opt-in “the approval surface exists for the act and not for the claim” (§8.3).

The correction. Grant the smallest set of tools the task needs, put a person's approval on anything irreversible, and treat any claim of an action as unverified until the action's own record shows it.

4.5 The state and artifact store: what it keeps

The survey's store "persists execution state and products, including conversation history, plans, scratchpads, checkpoints, logs, traces, diffs, memory records, generated files, and task artifacts." This is where a model's apparent memory lives. MemGPT made the idea explicit, moving information between the limited context window and external storage the way an operating system manages memory (Packer et al., 2023), and Anthropic's guidance calls the pattern "Structured note-taking, or agentic memory": "the agent regularly writes notes persisted to memory outside of the context window" that are pulled back in later (Anthropic, 2025a). What the store keeps, the model later reads as knowledge. OpenAI's account of the GPT-4o episode in background paper 7 notes that "in some cases, user memory contributes to exacerbating the effects of sycophancy", while adding that it did not have evidence that memory broadly increased it (OpenAI, 2025).

From the logs. The series again supplies the case. An author's first name was carried across several drafts from the practice's own notes rather than read from the byline, until the citation check read the byline. The practice's memory entries now carry an instruction to verify them before use, and its handoff files point to the record rather than standing in for it.

The correction. Keep the record machine-written where possible (§6 of the working paper is the argument), treat stored notes as leads to check, and keep what an agent writes about itself separate from what the machinery records about it.

4.6 The verification and governance layer: who checks

The last component "checks, constrains, and repairs execution through tests, assertions, verifier models, sandbox policies, permission gates, rollback, retry, budget control, safety constraints, and audit traces." Background paper 7 explains why this job cannot be left to the model: its training pays for agreement and for passing checks. The working paper adds the design rule that decides whether a check works: "The variable is whether the verdict depends on evidence the claimant could not have authored or talked its way around" (§8.3). A check that reads the claimant's own account, whether it is deterministic or a model, rots.

A harness can also change what a trained habit costs the model. Gomez (2026) gave coding agents facing defective tests a structured way to report the defect, and separately a plain policy against gaming the tests. Across eight frontier models from five families, reward hacking fell from 23.6% with neither, to 15.0% with the reporting tool alone, 9.7% with the policy alone, and 5.3% with both, and nearly every escalation involved no hacking. The training that taught the models to pass checks was untouched; the harness gave them a better move. Figure 5 in §7 draws the four conditions.

From the logs. Every citation in this series was checked at its source before publication, which is what caught the broadcast date of §4.1 and the byline of §4.5; a count of the blog's posts was corrected from

seven to eight before it went out, when the page itself was counted. The practice's own attempt at an automated checker is the cautionary case: a small open-weight auditor, shown claims and their evidence, never flagged a false one, and under a better prompt flagged true and false claims at the same rate (§9 of the working paper; background paper 7, §7).

The correction. Checks that rest on evidence the claimant could not author: test output, the page itself, the record. And a place for the model to say something is wrong that is cheaper than hiding it.

5 Same model, different harness

If the harness decides as much as §4 suggests, the same model should perform very differently in different harnesses. It does. Guo et al. (2026) compile published results in which the model is held fixed and only the harness changes. On Terminal-Bench 2.0, Claude Opus 4.6 “ranges from 58.0% with Claude Code to 76.4% with Meta-Harness”, GPT-5.3-Codex from 64.7% to 78.4%, and Gemini 3.1 Pro from 59.4% to 80.2%. On WebArena, GPT-4o “ranges from 13.1% in the model-only baseline to 54.6% with WebOperator”. On SWE-bench Verified, source-reported harnesses take GPT-4o from 23.2% to 38.8%. Figure 3 draws these ranges.

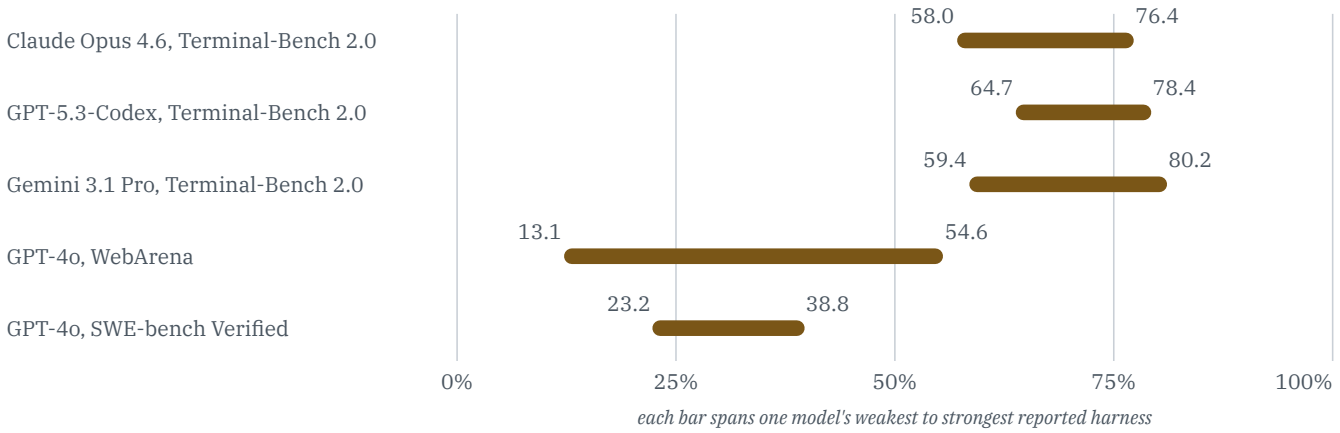

Figure 3. The same model's reported score in its weakest and its strongest harness, as compiled by Guo et al. (2026, §7) from published and leaderboard results. Benchmarks differ, so the bars compare spans, not levels. Only the harness varies within a bar. **Measured, as compiled.**

Controlled studies point the same way. Yang et al. (2024) built SWE-agent around the argument that language models are a new kind of user that needs its own interface, and ablated that interface with the model fixed. With GPT-4 Turbo on SWE-bench Lite, the full interface solved 18.0% of tasks; removing its editing tool dropped that to 10.3%, and replacing its compact file viewer or its search tools with plainer versions cost several points each (Figure 4). Xia et al. (2024) went the other way and removed the agent loop altogether: a fixed three-phase pipeline of localization, repair and patch validation, which never lets the model decide its next action, solved 32.00% of SWE-bench Lite with GPT-4o at an average cost of $0.70, the best result among open-source approaches at the time. And Riegler and

Strümke (2026) set instances of a 1.2-billion-parameter open-weight model against a program with nine planted vulnerabilities: with a hand-built seed corpus, pattern detection and crash classification in the harness, the system found all nine; with those components switched off, the same model found none by crash verification and two by citation. Their conclusion is that capability of this kind "is a property of the system", and that on their target "the recall belongs to the scaffold."

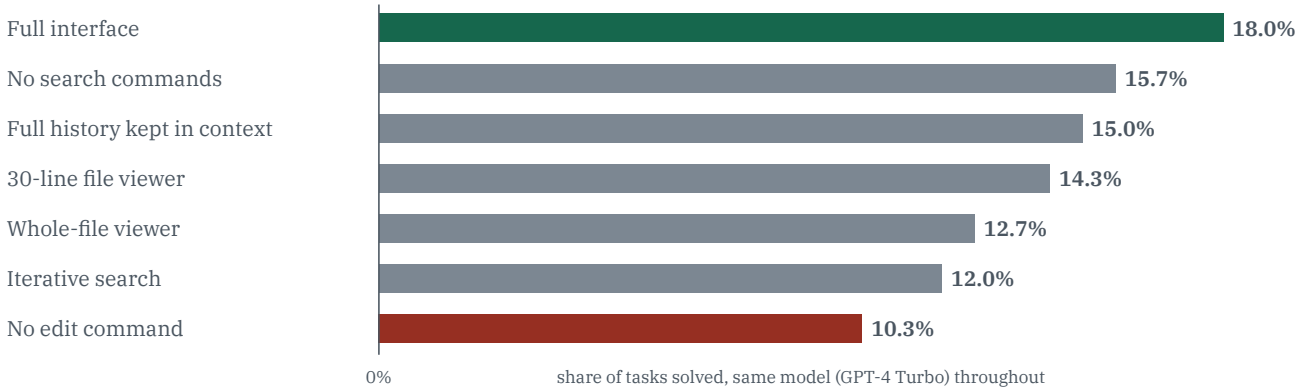

Figure 4. SWE-agent's ablations on SWE-bench Lite with the model held fixed (Yang et al., 2024). Changing how the agent searches, views files, keeps its history or edits moved the share of tasks solved from 18.0% down to as little as 10.3%. **Measured.**

The evidence has a second half, and the argument is not complete without it. The model matters too: within a single harness, Guo et al. report a move from 49.0% to 73.2% on SWE-bench Verified from a change of model alone. And the harness does not matter equally for everything. In a pre-registered study of six frontier models across four configurations, Gringras (2026) found that the choice of harness architecture explained only 0.4% of the variance in measured safety scores, with the choice of benchmark explaining about 45 times as much; within that small average sat large swings for particular models, such as Claude Opus 4.6 losing 16.8 percentage points and Llama 4 Maverick gaining 18.8 under the same configuration on one benchmark. Kapoor et al. (2025), running 21,730 agent runs across nine models and nine benchmarks, found that the choice of harness could decide both cost and accuracy, and that more reasoning effort did not improve accuracy in 21 of the 36 comparisons they could make. The conclusion this paper draws is the modest one: a published score is a score for a model in a harness, and a model bought on a score will behave like that score only in a harness like the one that produced it.

6 Open and closed weights, seen from the harness

Background paper 7 separates models by who holds the weights. From the harness side the distinction looks like this.

With a closed-weight model, rented through an API, the harness is everything the user controls. The model returns text rather than its probabilities, so nothing can steer it from inside; it can be changed by

its vendor, as the GPT-4o episode showed; and every habit its training left can only be worked around, in what the harness shows it, lets it do, lets it stop on, and checks. The harness is the user's whole lever.

With an open-weight model, the harness can reach further in. It controls how tokens are sampled, can read the model's probabilities, and can steer one model with a smaller adapted one, which the working paper notes "requires access to logits and therefore open weights" (§8.4). It can run entirely on the user's own hardware: every result in Riegler and Strümke's study was produced "with open-weights models running locally on consumer hardware, with no API calls to frontier providers, no data leaving the machine, and no recurring cost." And the model underneath can be trained further, which background paper 7 (§9) describes as the second lever.

Either way the harness decides the working day. An open-weight model adds a lever; it does not remove the need for the first one.

7 Where the corrections live

Background paper 7 ends with a map from each training stage to the habit it leaves and to where the correction can act. For a closed-weight model, every correction on that map is a harness correction. Table 1 lists them with what each costs and what it cannot do, and Figure 5 draws the one intervention in the table measured across many frontier models.

Table 1. The habits of background paper 7 and the harness corrections for each. "Cannot" is the honest limit: what the correction leaves untouched.

Habit (paper 7)	Harness correction	What it costs	What it cannot do
Common misconceptions; plausible guesses on rare facts	Load the sources; require claims to cite them; check each citation at its source; credit "I don't know"	Retrieval, and a check per claim	Correct what the sources themselves get wrong
The expert voice with nothing behind it	Ask for the evidence and a stated confidence; treat fluency as no evidence at all	The reader's attention	Flag a confident error that cites nothing checkable
Agreement; folding under pressure	Challenges shaped as evidence, citing a record, in an advisor's register; a decider anchored on the logs rather than on another opinion (§8.2 of the working paper)	Building and keeping the record	Distinguish a correction from a fold without an independent signal of what was right (H3)
Passing the checker	Checks the agent cannot edit or author; held-out tests; a way to report a broken test, and a stated rule against gaming it (Gomez, 2026)	Test infrastructure, and reviewing what gets reported	Cover a dimension that no check evaluates

Habit (paper 7)	Harness correction	What it costs	What it cannot do
Stopping at the first defensible answer	A stopping rule that requires evidence; a challenge at the stopping moment (§8)	Latency, and a challenge budget	Guarantee that the answer, once checked, is right
An account in place of a record	A record written by the machinery as work happens, not by the agent about itself (§6 of the working paper)	Instrumentation	Replace the judgment of what the record means

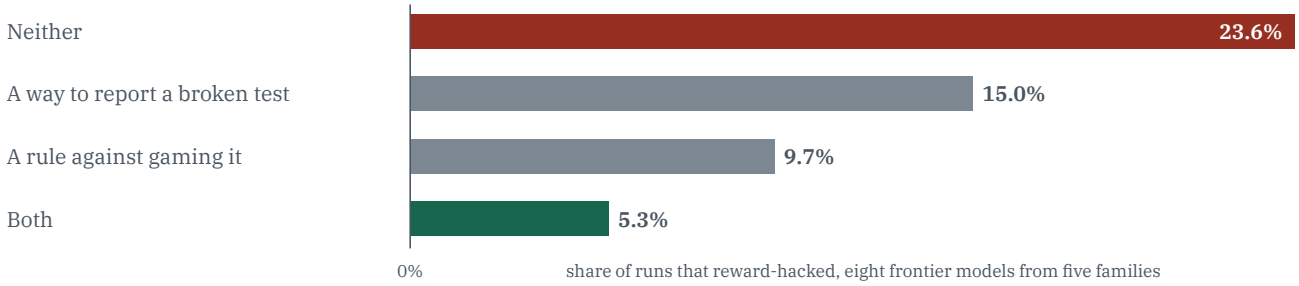

Figure 5. Reward hacking by coding agents facing defective tests, under a two-by-two of harness interventions (Gomez, 2026). The models' training was the same in every cell; only the harness changed. **Measured.**

One result bounds how far any of this reaches. In the research setting of background paper 7 (§5.4), safety training delivered through chat-like prompts removed misaligned behavior on chat-like evaluations while it persisted on agentic tasks (MacDiarmid et al., 2025). A correction installed in one part of a system does not automatically carry to the parts that act. Harness corrections are the same: each covers the path it sits on.

8 From the logs: the practice's harness and its counterweight

The practice this series describes runs its work through agents built on a closed frontier model family, under a harness of hooks, shared logs and automated checks, directed by an operator who approves every consequential change (§2.2 of the working paper). Described by what it does rather than by its parts, that harness loads its governing rules into an agent's context before the agent's first token; checks each turn's end for promised actions not taken and claims not verified; routes every change to production through the operator's approval; refuses command shapes that have failed silently before; treats its stored notes as leads to verify; records a compaction whenever it happens and expects a reload before the agent acts on what it lost; and writes its record by machinery as work happens rather than by asking agents to report on themselves. It still misses. Section 4 gives four misses from the writing of this series alone, each caught at a source or by the operator, and the working paper's census finds the operator supplying corrective signal on something near half of what he types (§3.1).

The counterweight program is the practice's attempt to move part of that correction from the operator into the harness. Background paper 7 (§9) gives its premise and is careful about its status: that the stop is early is an inference from observation, inside the practice and outside it, and that the cause is a pull toward acceptance in weights the practice cannot touch, always on while the operator's correction is present only when he is, is a hypothesis the program exists to test. Here is its shape as a harness component.

Two controllers. A fast controller acts within a turn: a challenge placed in the agent's context at the moment it would stop, citing a record the agent can read. A slow controller acts overnight: a consolidation of the day's corrections into the material the next day's challenges are drawn from (§8.4 of the working paper, H2). In Guo et al.'s terms, the fast controller sits in the control loop and the context manager, and the slow one in the state store.

Separate bodies. The model doing the work is not the model that challenges it, and the model that writes a challenge is not the model that judges one. In the first build both the challenger and the auditor are open-weight models from different families, running on the practice's own hardware (§8.5 of the working paper). The current design adds a third component, a verifier that checks a claim against the sources it cites: an open-source literature-checking harness, PaperQA2, driven by a model from yet another family. That component is designed and not built. The reason for separate families is the working paper's reading of Knight and Leveson (1986): checkers built alike fail alike, and two instruments that share a blind spot are one instrument. Two results on language models point the same way: evaluators recognize and favor their own generations (Panickssery, Bowman & Feng, 2024), and when a generator and an evaluator share an underlying model, reward hacking can appear in context with no training at all (Pan, He, Bowman & Feng, 2024). One of the checking models is preferred fully open for exactly this reason: with its training data published, whether it is independent of the other can be audited rather than assumed.

Why this lives in the harness. Every part of that design is one of the harness jobs of §4. The challenge enters through the context manager; its timing belongs to the control loop; its evidence comes from the state store; its judge is the verification layer. None of it requires changing the working model, which is the point: the practice cannot change the working model.

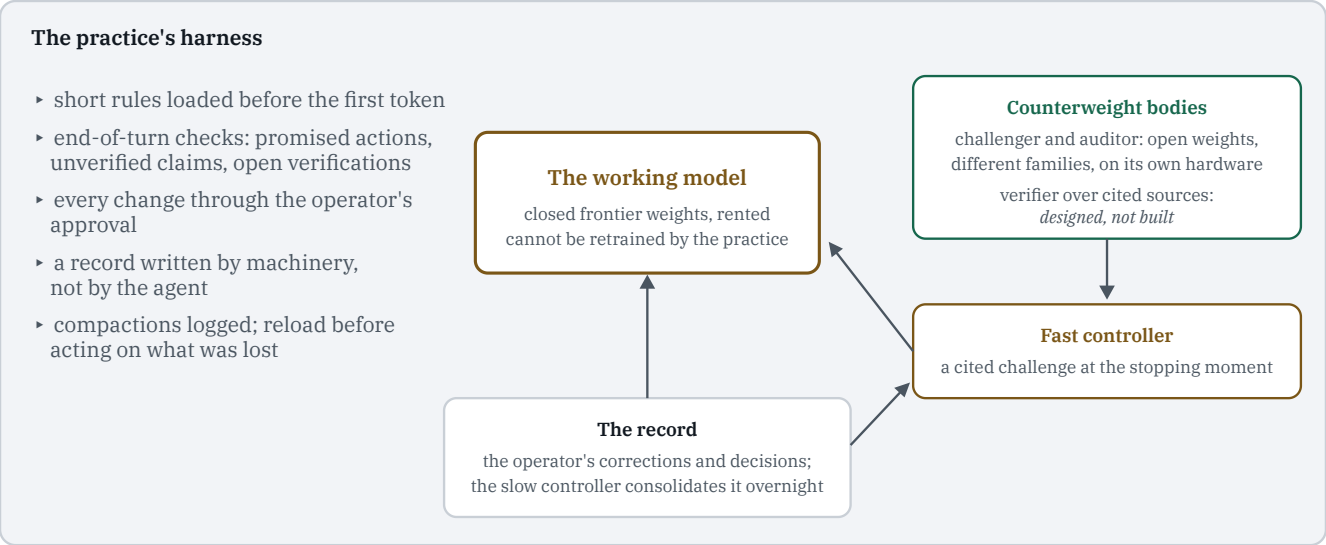

Figure 6. The practice's harness, described by function, and the components the counterweight program adds to it. The challenge engine that produces the fast controller's challenges is built and not deployed; the delivery switch is built and unarmed; the verifier over cited sources is designed and not built (§8.5, §9 and §10.9 of the working paper; this section). **Schematic; build status shown.**

Where it stands. The challenge engine was implemented and merged on 11 September 2026 and is not deployed; its delivery switch is built and deliberately unarmed; and its first calibration failed, which is why neither auditor is deployed on those numbers (§8.5, §9 and §10.9 of the working paper). The program's primary study has a pre-registered null that closes it. What would make this component worth having is not that it is built, but that challenges it delivers turn out to be corrections rather than folds, measured against a signal of correctness the challenged agent cannot author. That measurement has not been made.

9 What you rent and what you own

The working paper observed that over 2026 the infrastructure beneath working agents, the orchestration, retries and state handling, “became a managed rental”, while the decisions it exists to execute did not: whose knowledge becomes the standing instructions, what result ends an experiment, and what “done” means for a given piece of work (§8.6). This paper adds one line to that account. The model is rented, and so, increasingly, is the plumbing. What is not rented is the harness configuration that carries a user's definition of right into the work: what it loads, what it lets the model touch, when it lets the model stop, and what checks the result, together with the record of every correction that definition has required.

For a closed-weight model, that makes the harness something other than an accessory to the product. For the work of the person using it, the harness is where the product is decided.

10 Limits

The comparative evidence is uneven. Many of the same-model comparisons in §5 are compiled from leaderboards whose settings differ in ways a compilation cannot fully control, as Guo et al. note of their own tables. The controlled studies are narrower: one model and one benchmark for SWE-agent's ablations, one target program for Riegler and Strümke, one kind of task for Gomez. Gringras (2026) is the reminder that the size of the harness effect depends on what is measured.

The taxonomy is a map. Guo et al. say themselves that the mapping from their components to the steps of the loop is many-to-many. The six jobs of §4 overlap, and a real failure often involves several.

The practice's description is qualitative. Section 8 describes the practice's harness by function and reports no rate at which its checks catch anything; the practice's record contains only errors that were caught, so it can support a share of catches with its window and never a catch rate (§4.1 of the working paper). The four misses in §4 were chosen because they illustrate the components, not sampled.

What would change the argument. Evidence that as models improve, a model run bare, or in a minimal harness, matches the best harnesses on agentic work would weaken the second claim of §1. Evidence that checks resting on evidence the claimant could not author fail to discriminate true claims from false ones would weaken the design rule of §4.6; the practice's own first auditor, which did not discriminate, is a caution rather than a counterexample, because it read the claimant's own window.

How to cite this part. Atkinson, B. (2026). *What Is a Harness? The Software Around the Model, and Why It Decides So Much.* Background paper 8 to *How do we use this?* Working paper, Wolfberg LLC.

Corrections, 28 September 2026. The primer is now four posts, in order: “What is an LLM?”, “What is a harness?”, “What is context?” and “What is ‘AI’?”, and this paper's references to it follow that order. Section 4.2 now reports current evidence on long contexts, as background paper 9 does: the 2023 finding that models use the middle of a long input worst is given with its year, beside a 2026 reproduction with current models that did not find it, and the 2024 figure on 32,000-token contexts is withdrawn. Section 8 now says that the rules loaded before the first token are the short ones; the practice's two long rulebooks have to be read again after each compaction, which background paper 9 measures.

Corrections, 25 September 2026. No figure changed. Two sentences drawn from §8.3 of the working paper now match its correction of the same date: CrewAI, one of the three frameworks surveyed, lets a task have a human review the agent's final answer (off by default), and its run ends on the words “Final Answer” rather than on a turn with no tool calls.

Disclosure. Drafting and literature synthesis were assisted by AI models (Claude, Anthropic) working under the author's direction; the author is responsible for the content. The works in the References were read in full, by AI reading agents working under the author's direction, and every quotation and number

this paper takes from them was then checked by the drafting model against the saved full text of its source. The works under Prior art are cited at the level of an established concept and its origin: each was verified for author, title, year and venue, none was read in full, and no numeric claim rests on any of them. Pages from laboratories, vendors and standards bodies were read at the linked page on 24 September 2026 and are current as of then, not permanent. Every arXiv identifier below was resolved against arXiv on 24 September 2026 with its title and first author matched; the three 2026 studies added in the correction of 28 September were read in full and resolved the same way on 27 September 2026, for background paper 9.

Competing interests. The author owns Wolfberg LLC, the practice studied. The practice's working model is from Anthropic's Claude family, the drafting assistance used the same family, and several of the sources are by Anthropic or about Anthropic's products.

Data availability. The practice's logs contain client work and personal records. They are private and are not offered for sale or sharing. Figures from the practice are the working paper's, reported as of the dates it gives.

REFERENCES

Atkinson, B. (2026a). *How Do We Use This? Findings from running an AI team on real work for six months, and a test that could prove them wrong.* Working paper, Wolfberg LLC. wolfberg.ai/papers/how-do-we-use-this

Atkinson, B. (2026b). *What Is an LLM? How a Language Model Is Made, and Where Its Habits Come From.* Background paper 7 to *How do we use this?* Wolfberg LLC. wolfberg.ai/papers/how-do-we-use-this/part-7

Atkinson, B. (2026c). *What Is Context? What a Model Can Hold, and What Happens When It Runs Out of Room.* Background paper 9 to *How do we use this?* Wolfberg LLC. wolfberg.ai/papers/how-do-we-use-this/part-9

Bousetouane, F. (2026). *AI agents do not fail alone: The context fails first.* arXiv:2607.14275. arxiv.org/abs/2607.14275

Cemri, M., Pan, M. Z., Yang, S., et al. (2025). *Why do multi-agent LLM systems fail?* arXiv:2503.13657. arxiv.org/abs/2503.13657

Gabín, J., Perez, A., & Parapar, J. (2026). *Lost in the evidence? Reproducing document position and context size effects in RAG.* In *Proceedings of SIGIR '26*. arXiv:2605.27105. arxiv.org/abs/2605.27105

Gomez, F. (2026). *Can escalation channels redirect reward hacking toward defect disclosure?* arXiv:2608.29460. arxiv.org/abs/2608.29460

Greshake, K., Abdelnabi, S., Mishra, S., et al. (2023). *Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection.* arXiv:2302.12173. arxiv.org/abs/2302.12173

Gringras, D. (2026). *Safety under scaffolding: How evaluation conditions shape measured safety.* arXiv:2603.10044. arxiv.org/abs/2603.10044

Guo, J., et al. (2026). *From question answering to task completion: A survey on agent system and harness design.* arXiv:2606.20683. arxiv.org/abs/2606.20683

Kapoor, S., et al. (2025). *Holistic Agent Leaderboard: The missing infrastructure for AI agent evaluation.* arXiv:2510.11977. arxiv.org/abs/2510.11977

Liu, N. F., Lin, K., Hewitt, J., et al. (2023). *Lost in the middle: How language models use long contexts.* arXiv:2307.03172. arxiv.org/abs/2307.03172

MacDiarmid, M., et al. (2025). *Natural emergent misalignment from reward hacking in production RL.* arXiv:2511.18397. arxiv.org/abs/2511.18397

Packer, C., Wooders, S., Lin, K., et al. (2023). *MemGPT: Towards LLMs as operating systems.* arXiv:2310.08560. arxiv.org/abs/2310.08560

Riegler, M. A., & Strümke, I. (2026). *AI security policy should assess systems, not only models.* arXiv:2605.09504. arxiv.org/abs/2605.09504

Wallace, E., Xiao, K., Leike, R., et al. (2024). *The instruction hierarchy: Training LLMs to prioritize privileged instructions.* arXiv:2404.13208. arxiv.org/abs/2404.13208

Xia, C. S., Deng, Y., Dunn, S., & Zhang, L. (2024). *Agentless: Demystifying LLM-based software engineering agents.* arXiv:2407.01489. arxiv.org/abs/2407.01489

Yang, J., Jimenez, C. E., Wettig, A., et al. (2024). *SWE-agent: Agent-computer interfaces enable automated software engineering.* arXiv:2405.15793. arxiv.org/abs/2405.15793

Zeng, W., Huang, Y., & He, J. (2026). *LOCA-bench: Benchmarking language agents under controllable and extreme context growth.* arXiv:2602.07962. arxiv.org/abs/2602.07962

Zhang, C., Cui, H., Huang, X., & Sang, J. (2026). *Positional failures in long-context LLMs: A blind spot in reasoning benchmarks.* arXiv:2605.23170. arxiv.org/abs/2605.23170

PRIOR ART (CITED AT CONCEPT LEVEL; VERIFIED FOR AUTHOR, TITLE, YEAR AND VENUE, NOT READ IN FULL)

Adler, S. (2025, October 2). *Practical tips for reducing chatbot psychosis.* Clear-Eyed AI. Cited as reported in the working paper (§1), where it was read in full. clear-eyed.ai

Knight, J. C., & Leveson, N. G. (1986). An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Transactions on Software Engineering*, SE-12(1), 96–109. Cited as reported in the working paper (§2.1). doi.org/10.1109/TSE.1986.6312924

Pan, J., He, H., Bowman, S. R., & Feng, S. (2024). *Spontaneous reward hacking in iterative self-refinement.* arXiv:2407.04549. Cited as reported in the working paper (§8.3). arxiv.org/abs/2407.04549

Panickssery, A., Bowman, S. R., & Feng, S. (2024). *LLM evaluators recognize and favor their own generations.* arXiv:2404.13076. Cited as reported in the working paper (§8.3). arxiv.org/abs/2404.13076

Yao, S., Zhao, J., Yu, D., et al. (2022). *ReAct: Synergizing reasoning and acting in language models.* arXiv:2210.03629. arxiv.org/abs/2210.03629

LABORATORY AND VENDOR PAGES, STANDARDS, AND REPORTS (READ AT THE LINKED PAGES, 24 SEPTEMBER 2026)

Anthropic. (2024, December 19). *Building effective agents.* anthropic.com/engineering/building-effective-agents

Anthropic. (2025a, September 29). *Effective context engineering for AI agents.* anthropic.com/engineering/effective-context-engineering-for-ai-agents

Anthropic. (2025b, November 26). *Effective harnesses for long-running agents.* anthropic.com/engineering/effective-harnesses-for-long-running-agents

Hong, K., Troynikov, A., & Huber, J. (2025, July). *Context rot: How increasing input tokens impacts LLM performance.* Chroma technical report. research.trychroma.com/context-rot

Model Context Protocol. (2025). *Specification, version 2025-06-18.* modelcontextprotocol.io/specification/2025-06-18

OpenAI. (2025, May 2). *Expanding on what we missed with sycophancy.* openai.com/index/expanding-on-sycophancy