AGENTIC MYOPIA: PROBING THE TEMPORAL AWARENESS OF AGENTS IN SIMULATED MULTI-TURN CODING SESSIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Coding agents increasingly operate over long, multi-turn sessions, in which a user and an agent collaborate on a large software task. As these sessions lengthen, agents are delegated increasing autonomy, completing substantial subtasks with limited user oversight. This delegation introduces a coordination cost: a user who intends to plan around, budget for, or intervene in the agent's work requires an accurate estimate of the work that remains. We term an agent's capacity to produce this estimate of remaining work (measured in outstanding turns, wall-clock time, and generated tokens) its **temporal awareness**. This capacity remains largely unexamined in the multi-turn setting in which coding agents are now deployed, where requirements emerge over the course of a session rather than being fixed at its start. We study coding agent temporal awareness in five frontier-hard software environments, in which a simulated user reveals task instruction to a coding agent incrementally. Using a carefully-designed fork-and-discard probing protocol to estimate five quantities of remaining work in a session, we find that frontier agents largely lack temporal awareness. Their answers are myopic: estimates of remaining work barely move as a session runs down. Given the full list of requirements still to come, agents size the remaining turns almost exactly, with the ratio of predicted to actual rising from 0.27 to 0.97. Curiously, the same list pushes wall-clock estimates to three times the truth. Token estimates climb only from 3% of the truth to about a quarter of it, so the two costs of the remaining work err in opposite directions. An agent that knows how much is left can still misjudge what it will cost, and coding agents cannot yet be trusted to budget the rest of a session.

1 INTRODUCTION

Somewhere right now a developer has handed a task to a coding agent and is deciding what to do with the next hour. They could start something else, or tell a reviewer that a change is coming and hope it does. They could take the task back if it looks to be going badly, and the later they do, the more of the hour is gone. Each is a scheduling decision, and all turn on the same unknown: how much of the work is left. Misjudge it and the cost is not a worse patch but a worse plan, work queued behind a job nowhere near done, or abandoned a turn before it would have finished. The question, then, is whether an agent can tell the person waiting how much is still to come. We call the capacity to produce a running estimate of remaining work **temporal awareness**, and we measure it in the units that developer would plan around: exchanges still to go, minutes, and tokens.

Nothing about capability guarantees this. Given a complete specification, a model can decompose it and count the parts left. However, multi-turn sessions do not supply a complete specification: a user states a requirement, sees the result, and states a second requirement that the first made visible, so the specification is assembled over the session rather than delivered at its start (Baumann et al., 2026; Tack et al., 2026; Laban et al., 2025). Estimating remaining work under these conditions means keeping a model of what this particular user still wants, a form of user mental modeling that recent work has begun to study (Zhou et al., 2026; Zhang et al., 2024). It is also not a question about what the model represents internally. Investigation into chain-of-thought traces show that agents may anticipate the next few user turns (Xu et al., 2026). Whether an agent can *report* how much of the session remains is a different claim.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Language models are known to handle time badly in isolation, on ordering, duration and arithmetic over dates (Ofengenden and Andriushchenko, 2026; Fatemi et al., 2025; Garikaparthi, 2026). Agents asked to forecast their own cost on a task handed to them in full forecast low (Bai et al., 2026; Lin et al., 2026). And interactive evaluation now supplies a user who reveals what they want over several turns (Vijayvargiya et al., 2026; Zhou et al., 2026; Raghavendra et al., 2026). The missing step is the combination: asking an agent, *while* its still working with a live user, how much of the session is left.

A single estimate made at the start concerns a fixed amount of work. A *running* estimate has to be revised as work is done and as the specification grows, and it is the running estimate that the developer waiting on a session uses. An agent can be good at the first and useless at the second. We therefore probe repeatedly through each session and ask not only whether an estimate is right on average but whether it *moves*, whether it comes down as the work does. Two obstacles have kept this from being measured. There is no referent: asking an agent how much work is left means something only if the true answer is known, and in an open-ended session it is not. And asking interferes: a question posed mid-session becomes part of the session, so a probe about remaining work may prompt the planning it is meant to observe. Both hazards are long established for verbal self-report in people (Nisbett and Wilson, 1977; Ericsson and Simon, 1980; Schwarz, 1999), and language models inherit the same sensitivity to paraphrase and reframing (Rupprecht et al., 2026; Singh et al., 2026).

We address both obstacles, the missing referent and the interference, by controlling the user rather than the probe. We use a simulated user which gradually reveals an ordered list of concrete requirements, with the next requirement raised only once the current one is settled. Simulated users are an established instrument for multi-turn evaluation (Naous et al., 2026; Zhu et al., 2026); ours supplies the referent, since remaining work becomes a count the simulator holds and a finished session recovers exactly. To ask without interfering, we rebuild the agent's context on a disposable branch, pose one question, record the answer, and discard the branch, so the retained session never contains the probe. We call this the **fork-and-discard** protocol. Because the branch is reconstructed rather than resumed, the same point in a session can be asked repeatedly, which permits controlled variation of both the question and the information the agent is given.

We study two primary research questions. **RQ1**: Can coding agents estimate how much work remains as a session proceeds? **RQ2**: Does supplying the information the agent lacks recover the estimate, and for which measures?

This work makes three contributions. First, we release a benchmark of multi-turn coding tasks paired with a preference-based user simulator that reveals requirements incrementally, built from single-turn tasks by a recipe that extends to other task families. Second, we introduce a fork-and-discard probing protocol that reads an agent's estimate of its remaining work without disturbing the session. Third, we use both to evaluate five coding agents, with three findings. Agents are myopic (RQ1): their estimates of remaining work barely move as a session runs down, with a calibration slope of $+0.02$ against the 1.0 that tracking would require. Supplying the full requirement list repairs the count (RQ2): the ratio of predicted to actual remaining turns rises from 0.27 to 0.97. It does not repair the cost. The same list pushes wall-clock estimates to three times the truth, while token estimates climb only from 3% of the truth to about a quarter of it, so the two costs of the remaining work err in opposite directions. An agent that knows how much is left can still misjudge what it will cost.

2 METHODS

2.1 SETUP

A session alternates between a simulated user and a coding agent working in a shared repository. At each step the user sends a message and the agent replies, editing files as it does so. Following Tack et al. (2026), the user has an explicit, structured intent rather than a persona. It holds an ordered list of requirements $\mathcal{R} = (r_1, \ldots, r_n)$. Each requirement is a concrete, verifiable property of the finished artifact, phrased as a user would state it. A requirement is *revealed* once the user has voiced it to the agent and *settled* once the user considers it complete.

2.2 CODING AGENTS

We evaluate five agents, each a model paired with the harness it is normally deployed in. Three are frontier: `gpt-5.6-sol` under Codex (OpenAI, 2025), and `claude-opus-5` and

`claude-sonnet-4.5` under Claude Code (Anthropic, 2025). Two are deliberately weaker, to separate temporal awareness from capability: `gpt-5-mini` under Codex and the open-weight `qwen3.5-27b` under OpenCode (Yang et al., 2025; Merrill et al., 2026). Each agent receives a short system prompt asking it to work with the user and follow their instructions. The prompt names no part of the task, so the agent learns the requirements only from the user's messages.

2.3 User Simulator

Simulated users are an established instrument for multi-turn evaluation (Naous et al., 2026; Zhu et al., 2026). Persona-based simulators, however, leave what the user wants implicit (Naous et al., 2026). Holding the requirement list as explicit state has two consequences: remaining work becomes a count with a known value, and requirements arrive during the session rather than at its start, which is the setting in which we expect estimation to fail.

The simulator reveals requirements sequentially. After each agent reply, it judges from the agent's message whether the outstanding requirement has been met. If so, the requirement is settled and the next one is voiced; otherwise the user presses the current one. A session ends when every requirement is settled or after $2n$ steps (Table 3). Reward is computed separately by the task verifier, which inspects the artifact, so a session can be fully settled and still score poorly. In each environment, user simulators are driven by `qwen3.5-27b`, and we investigate results using stronger agents to run the simulator in Appendix F.5.

2.4 Tasks

We use five terminal-based coding tasks in the Harbor format (Harbor Framework Team, 2026) of Terminal Bench 2.0 (Merrill et al., 2026), each a sealed container with a reference solution and a verifier. They carry 8 to 18 requirements each (median 12), span JavaScript, Python and Rust (Table 3), and are modeled on requests observed in real coding-agent sessions (Baumann et al., 2026).

Each task is converted from single-turn to multi-turn form in three steps. We first decompose the specification into requirements, each bound to a disjoint set of verifier rubrics, so reward is the fraction of requirements satisfied. We then seed the workspace with a skeleton of the reference solution in which every function body raises an error when called, so the agent can discover the interface by reading the code and an unedited workspace scores zero. We then validate each binding by ablation: removing a behaviour from the reference solution must fail exactly the rubrics bound to it. Each task also passes eight further checks, including that the reference solution scores 1.0 and that the agent's container contains neither the reference solution nor the verifier (Appendix A).

2.5 Probing

We define a probe as a question q posed to a subject, the user simulator or the agent, at step t and before that subject takes its turn. The subject answers by conditioning on its context c_t at that step and returning an indication $y \sim \pi(\cdot \mid c_t, q)$, from which we read a latent quantity such as its appraisal of progress. We probe by following the fork-and-discard protocol (Algorithm 1 in Appendix B).

The fork-and-discard protocol reconstructs agent context from the session state, which allows us roll out a session once and probe it afterward. At a chosen step we freeze the simulator's conditioning and reconstruct the agent's context, then replay that frozen state on a throwaway branch that never re-enters the live rollout. On the branch we fire the probe battery. Because the retained interaction never sees the probe, asking cannot change what the agent does next.

At every step we probe for four quantities: user turns remaining, agent turns remaining, wall-clock minutes remaining, and tokens remaining to be generated. All are counted from the point at which the probe is posed, after the user's message and before the agent's reply, and their true values are read from the completed session. Exact probe wordings are given in Appendix C.

We primarily are interested in whether agents can accurately predict remaining work under *local* context, in which the agent is given only the current state of the rollout. To find an upper bound on how well agents might predict remaining work, we also probu under *global* context, in which we prefix the full requirement list given to the simulated user and the user's current position in it.

Table 1: **Global context fixes the count and not the cost.** Median over the sessions probed in both conditions; ideal is 1.0 for slope and O:E. Skill is against a leave-one-out constant baseline (Section 2.6).

		Calibration slope		O:E ratio		Skill	
Measure	n	local	global	local	global	local	global
User turns	99	$+0.02$	$\mathbf{+0.85}$	0.27	**0.97**	-0.35	$+0.33$
Agent turns	99	$+0.02$	$\mathbf{+0.81}$	0.25	**0.98**	-0.36	$+0.26$
Wall clock	99	$+0.11$	$+2.24$	0.94	2.99	-0.02	-0.96
Tokens	99	$+0.00$	$+0.16$	0.03	0.23	-0.36	-0.22

Comparing pairs of local and global probe responses, we measure how well agents predict remaining work in both realistic, information-poor scenarios and in information-rich contexts.

2.6 Scoring

We summarise each session with two statistics from forecast evaluation (Mincer and Zarnowitz, 1969; Steyerberg et al., 2010), computed against that session's own ground truth. The *calibration slope* is $\beta = \mathrm{Cov}(\hat{y}, x)/\,\mathrm{Cov}(y, x)$, where $\hat{y}$ is the estimate, y the true value and x normalised progress through the session; $\beta = 1$ indicates an estimate that falls as the work does and $\beta = 0$ one that does not change. The *O:E ratio* is the mean estimate divided by the mean truth, with 1 indicating the correct level. We also report *skill*, one minus the ratio of the estimate's mean absolute error to that of a constant baseline predicting the median truth of the held-out sessions.

3 Experimental Results

We report 100 sessions: five tasks, each run four times by each of the five agents of Section 2.2. At every step the five-question battery is posed to both the agent and the simulator, under local and under global context, and each answer is scored against the true value read from the finished session (Section 2.6). Because the agent and the simulator are both sampled, the same task produces a different session each time, and sessions are re-rolled until they pass fixed validity checks, chief among them that the agent's container cannot reach the reference solution or the verifier (Appendix E).

3.1 Agentic Myopia

None of the five agents tracks how much of a session remains. As the true remaining work falls from thirteen exchanges to zero, the median estimate holds near 2: an agent one exchange from finishing and an agent with most of the session ahead of it report nearly the same number. No measure has a calibration slope distinguishable from zero (Table 1, local columns); for the two turn measures it is $+0.02$ against the 1.0 that tracking would require, with rank correlation $+0.09$. The count measures also sit far below the truth, at O:E 0.27 and 0.25, so agents under-state the number of exchanges remaining by roughly a factor of four while being unresponsive to progress. Wall clock is the exception in level, at O:E 0.94: agents land on roughly the right number of minutes with no sense of when those minutes fall. Token estimates sit at 3% of the truth (O:E 0.03).

Mean absolute error does shrink toward the end of a session. The decline is arithmetic rather than calibration: the truth descends toward a prediction that does not move, so the two converge whatever the agent believes. Scored against the constant baseline, agents have negative skill on all five measures, -0.35 on remaining user turns and -0.36 on remaining agent turns.

The flat median is not an average over sessions that behave differently. We label each session by its two statistics: a *Flat Prediction* if its calibration slope is below 0.30, an *Over-Prediction* if it responds but its O:E ratio exceeds 1.50, and *Loose Tracking* otherwise (the shares are stable across cutoffs; Appendix D.2). Without global context, 73% of sessions are Flat Predictions and 18% track (Appendix D.2). The flat median is what nearly three quarters of sessions individually do.

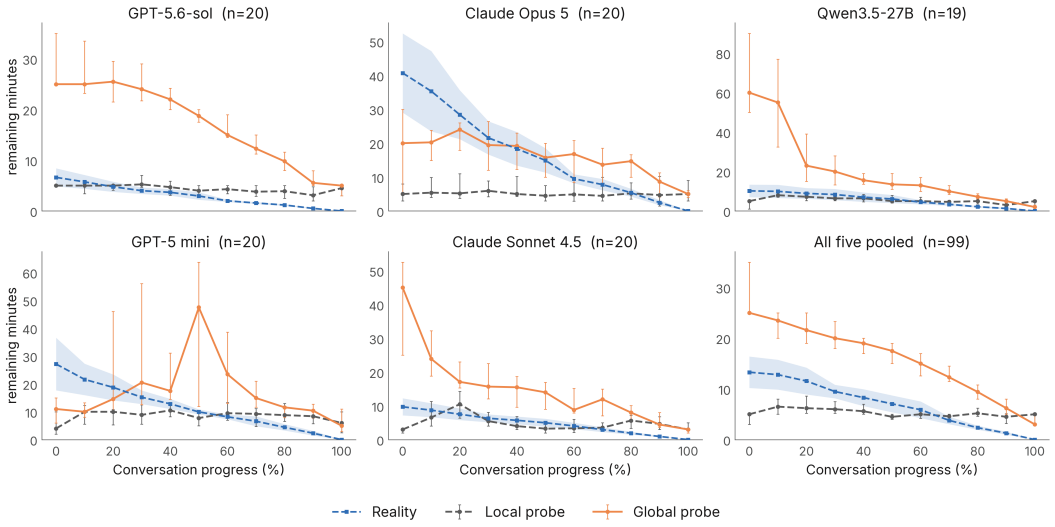


Figure 1: **No agent tracks remaining time, and global context makes four of five worse.** Remaining wall clock against conversation progress, one panel per agent and a pooled panel, over the 99 sessions probed in both conditions. Medians with bootstrap 95% intervals; the shaded band is the interval on the truth. Every agent is flat without global context (grey) and above the truth with it (orange).

3.2 Global Context Fixes the Count, Not the Cost

A flat estimate could reflect an inability to reason about the trajectory of a session, or the absence of the relevant information from the agent's context. Global context supplies the complete requirement list and the current position, with the question otherwise identical (Section 2.5). Because each probe is a discarded fork of the same frozen state, the two conditions are the same session asked twice, and we report the 99 sessions probed in both.

Global context calibrates the count measures and not the rate measures (Table 1, Figure 3). Remaining requirements, the measure the list states most directly, recovers most completely: the slope reaches $+1.00$ and skill $+0.68$. The turn measures come close, with slopes of $+0.85$ and $+0.81$, O:E of 0.97 and 0.98, and skill turning positive. Wall clock moves the wrong way. Its slope over-shoots to $+2.24$ while its O:E rises to 2.99, so the estimate begins to fall as the work falls, from three times too high, and skill drops from -0.02 to -0.96. Tokens move least, from 3% of the truth to 23% (O:E 0.03 to 0.23), so the two costs of the remaining work err in opposite directions: minutes are over-stated and tokens under-stated. Supplying the list tells the agent how many things remain, and agents use it. It does not tell them how long a thing takes or what it costs, and on those measures the added scope makes the estimate worse.

The per-agent picture is the same (Figure 1). Given global context, every agent's wall-clock estimate rises above the truth rather than onto it, and all five remain negative on skill. `gpt-5.6-sol` is the extreme case, its slope rising to $+3.94$ and its skill falling from -0.30 to -6.85; `claude-sonnet-4.5`, `qwen3.5-27b` and `gpt-5-mini` fall to -2.10, -1.88 and -1.31. Only `claude-opus-5` improves, from -0.11 to -0.02. At the session level, global context redirects the error rather than removing it (Appendix D.2). Flat Predictions fall from 73% to 21%, and the share that tracks falls as well, from 18% to 14%; Over-Prediction grows from 9% to 65%. Because the sessions are paired we can say where each one went: of the 72 that were flat, 41 become Over-Predictions and 11 begin to track, while 14 of the 18 that tracked without the list stop doing so once it is supplied.

The recovery of the count measures also rests on a convention. Under *batched reveal* the simulator releases three requirements per turn on a fixed schedule and does not wait for settling, which compresses a session from a median of 13 turns to 5 while leaving reward broadly unchanged. Given

Table 2: **Task success by agent and task**, mean reward over four repeats per cell (100 sessions). Reward is the fraction of bound per-requirement rubrics passed. The spread is what makes the calibration result interpretable: agents differ by more than a factor of two overall, and tasks by more than a factor of four, yet estimation error is uncorrelated with either (Section 3.3). Note in particular that the weakest agent here is the best calibrated, which is the opposite of what a capability explanation predicts.

Agent	Task 1	Task 2	Task 3	Task 4	Task 5	Mean
GPT-5.6-sol	0.82	0.90	0.40	0.91	0.06	0.62
Claude Opus 5	0.84	0.92	0.56	0.59	0.12	0.61
Claude Sonnet 4.5	0.83	0.51	0.14	0.88	0.34	0.54
GPT-5 mini	0.67	0.87	0.11	0.71	0.14	0.50
Qwen3.5-27B	0.39	0.21	0.11	0.49	0.20	0.28
Mean	0.71	0.68	0.26	0.72	0.17	0.51

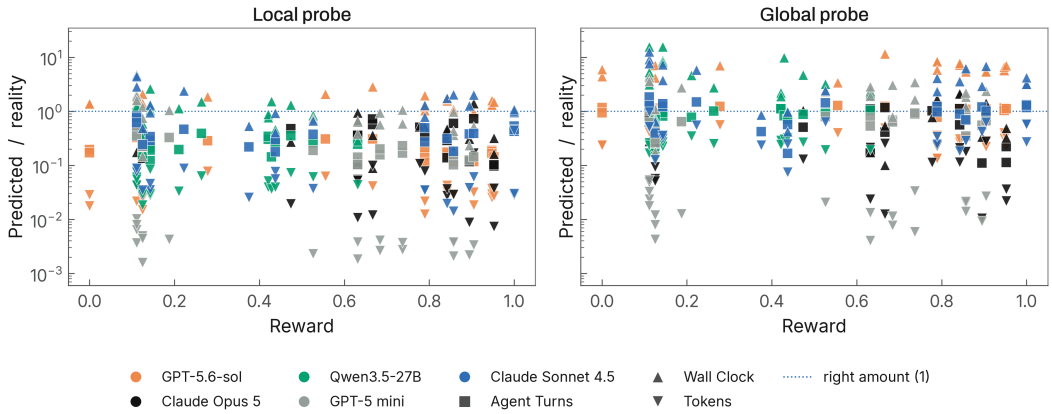


Figure 2: **How well a session goes says nothing about how well it is estimated.** Predicted work divided by actual work, one point per session and measure, against the reward that session earned. Marker shape is the measure, colour the agent, and the dotted line is a correct estimate. If the flat estimate were a symptom of a session going badly, the cloud would slope towards the line as reward rises. It does not, in either condition: the correlation between reward and $\log_{10}$ predicted-over-actual is -0.03 without the requirement list and -0.02 with it, over 252 and 295 session-measures. The three bands are the three measures, and they stay separated across the whole range of reward.

the same global context, the estimate over-shoots by roughly a factor of two, O:E 2.00 and slope $+1.91$, against 1.00 and $+0.92$ under sequential reveal (Appendix D).

The agent converts a count of outstanding requirements into a count of remaining turns at one per turn, which is right when the user behaves as ours does by default and wrong as soon as the user batches. What looks like calibration is in part an assumption our default simulator happens to satisfy.

3.3 Ruling Out Alternative Explanations

Four other readings of the flat estimate are available, and one condition closes each.

Never told that a requirement list exists, an agent may read "how many more messages?" as concerning the exchange currently under discussion, for which one or two is correct; 92% of local answers for remaining user turns are 0, 1, 2 or 3. Under *scope disclosure* the probe states that the user is working from an ordered list of requirements not yet stated and that the question concerns the entire remaining session, revealing neither the list, nor its length, nor the position (Appendix C). Agents roughly double their answer, the O:E for user turns rising from 0.27 to 0.51 and for agent turns from 0.25 to 0.53, so the framing is understood and acted on. The slope moves from $+0.02$

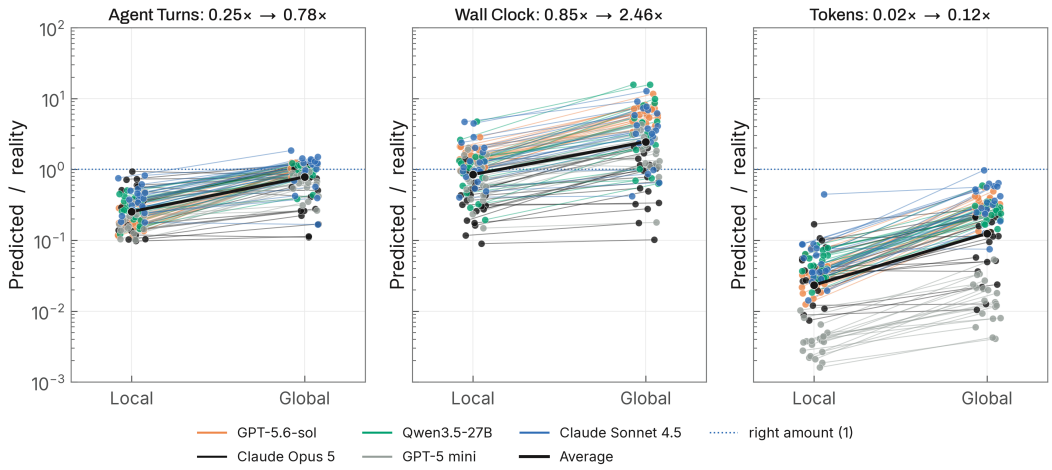


Figure 3: **If told what remains, agents fix the count and overshoot the cost.** Every session is probed both locally (session alone) and globally (told the full requirement list). Supplying the list moves agent turns from $0.25\times$ to $0.78\times$, close to right; the same information pushes wall clock from $0.85\times$ past the line to $2.46\times$, overshooting; and it lifts tokens only from $0.02\times$ to $0.12\times$, still roughly eight times too low. Knowing how much is left is not knowing what it will cost.

to $+0.05$, where global context takes it to $+0.85$. Wall clock over-shoots on framing alone: its O:E jumps from 0.96 to 2.62, most of the distance to the 3.04 that the actual list produces, while the slope stays near zero. The over-statement of Section 3.2 comes from learning that the session is longer than it looked, not from reading a long list.

Nor is the failure a property of scale or vendor. Without global context no agent beats the constant baseline on wall clock: $+0.00$ for `qwen3.5-27b`, -0.11 for `claude-opus-5`, -0.20 for `gpt-5-mini`, -0.30 for `gpt-5.6-sol` and -0.30 for `claude-sonnet-4.5` (Figure 1). Table 2 reports how well each agent does the work, and the two orderings are close to reversed. `qwen3.5-27b` solves least of the task, mean reward 0.28 against 0.62 for `gpt-5.6-sol`, and is the only agent whose local wall-clock estimate is not worse than the baseline. If a flat estimate were what an agent produces when it cannot do the work, the weakest agent would be the worst estimator.

A flat estimate could also be what a struggling session looks like. Figure 2 plots each session's estimate against the reward it earned, and the two are unrelated: the correlation between reward and $\log_{10}$ predicted-over-actual is -0.03 without global context and -0.02 with it, over 252 and 295 session-measures. Sessions that pass nearly every rubric are no better calibrated than sessions that pass almost none.

The remaining checks are on the instrument and are reported in Appendix F. Six rewordings move the slope by at most 0.28 (Section F.1); resampling one wording gives a standard deviation of 0.32 against an effect size of roughly 6 (Section F.2); rebuilding tool calls in full moves the slope by a median of $+0.03$; showing the agent its own earlier answers is a precise null; and a frontier simulator from a different model family changes the sessions without changing calibration.

4 Related Work

4.1 Temporal Reasoning in Language Models

Language models handle time poorly in ways that are now well documented. Benchmarks isolating ordering, duration and arithmetic over dates find persistent failures (Fatemi et al., 2025; Chu et al., 2024; Wang and Zhao, 2024; Tan et al., 2023), and the area is large enough to survey (Piryani et al., 2026). Inside a running task, agents are insensitive to elapsed time and to deadlines (Cheng et al., 2026; Sehgal et al., 2026; Ma et al., 2026). *Probing* names two activities in the LLM literature.

Reading a quantity off internal activations (Alain and Bengio, 2016) lets Silva et al. (2026) predict edit outcomes around twenty-five steps ahead in a coding agent, and Xu et al. (2026) recover latent plan structure in chain-of-thought the same way, so something in the model anticipates what is coming. We do the other thing, posing a question at the interface and reading the answer, and our claim concerns what an agent can report rather than what its activations contain.

4.2 Estimating Effort

The length of task an agent can complete has become a headline measure of progress (Kwa et al., 2025), but it is measured from outside. Asked directly, agents misjudge their own cost in a direction that depends on the unit. Wall-clock forecasts run high, by four to seven times on work that completes in seconds (Garikaparthi, 2026) and by three to seven times for Claude Code and Codex across eighteen benchmarks (Ofengenden and Andriushchenko, 2026). The latter study also finds forecasts close to a fixed prior, with a log-log slope near 0.2 against actual runtime, and no improvement when the agent is given more information. Token consumption, by contrast, is underestimated (Bai et al., 2026; Lin et al., 2026). Related work asks whether a model can hold to a budget it is given (Chen et al., 2026; Li et al., 2026) or predict difficulty before an attempt (Krsteski and Meyer, 2026; Shetty et al., 2026). In all of this work the task is given up front, so the quantity being forecast is fixed when it is forecast. Here the requirement list arrives a piece at a time, and the thing being estimated is still growing while the agent estimates it. Once our agents are given the list, they err in the same two directions (Section 3.2).

The calibration literature asks models what they know (Kadavath et al., 2022; Tian et al., 2023; Binder et al., 2025), a line now surveyed under the heading of metacognition (Liu et al., 2026a), and has measured agentic overconfidence directly (Kaddour et al., 2026). That work calibrates a claim the model has already made. We calibrate a claim about work not yet done, which is why our statistic is a within-session slope rather than a reliability diagram.

4.3 Multi-Turn Sessions with a Simulated User

User simulation trades controllability against realism. Controllable simulators condition a language model on an explicit persona or goal, which makes repeated agent evaluation possible (Tennenholtz et al., 2026; Yao et al., 2025; Barres et al., 2026), while realism-oriented work grounds simulated users in traces of real interaction (Baumann et al., 2026). Realism has proved hard to get: simulators built from helpful-assistant models diverge from human users, with stronger assistants making worse simulators (Naous et al., 2026; Dou et al., 2025), and they miscalibrate agent success (Seshadri et al., 2026). Persona conditioning, the usual remedy, does little at the individual level (Hu and Collier, 2024; Taday Morocho et al., 2026). Our simulator sits at the controllable end, closest in spirit to agenda-based simulation (Schatzmann et al., 2007), where the user's goal and pending acts are held as explicit state. Its job is to deliver a fixed list in a controlled order, which asks less than reproducing a person. One documented bias has a clear direction, and it runs against our result: simulated users are easier interlocutors than real ones (Liu et al., 2026c), and an easier user should make estimates better calibrated, not worse.

In software engineering, evaluation remains dominated by terminal outcomes such as whether a patch passes its tests (Wang et al., 2026b), though benchmarks increasingly include a user who withholds, clarifies or changes what they want (Raghavendra et al., 2026; Vijayvargiya et al., 2026; Zhou et al., 2026; Jin et al., 2026). Agents degrade when a specification arrives over several turns (Laban et al., 2025; Tack et al., 2026; Liu et al., 2026b) and often fail to ask when asking is warranted (Edwards and Schuster, 2026). Closest to us, Wang et al. (2026a) ask whether agents report task progress reliably while executing, as a single fraction on τ^2-bench, whose requirements are fixed in advance, and conclude that frameworks should not gate control flow on a model's own account of where it is. We ask for five separately verifiable quantities under incremental reveal and contrast the answers with and without the missing list; two instruments finding self-reported progress unreliable strengthens both. Replaying a saved session and branching from a frozen state has precedent (Nakash et al., 2026; Karpov, 2026); what we add is an exact referent for each branch.

5 Conclusion

We asked whether coding agents can tell how much work remains in a session whose requirements arrive one at a time. A simulated user that holds those requirements as an ordered list gives the question a true answer at every step, and fork-and-discard probing asks it there without the probe ever entering the session. The answer is that they cannot. Across five tasks and five agents of widely varying capability, estimates of remaining work barely move: the calibration slope on the turn measures is $+0.02$, against the 1.0 that tracking would require, and no rewording of the question lifts any slope above $+0.29$. A frontier simulator from another model family, tested on two tasks and two agents, changes the sessions but not the estimates, and the weakest agent is the best calibrated, so this is not a failure that capability has already fixed. Whatever agents represent internally, what they report does not follow the work that remains.

Supplying the full requirement list separates two capacities that are easy to conflate. Agents use the list to count what remains, partly by assuming one requirement per turn, a convention that a user who batches requests would break. They do not use it to price what remains, and their estimates of time and tokens then miss in opposite directions. One caution from the flat result applies beyond this paper. A quantity that falls to zero by construction flatters a constant predictor, since the error shrinks as the truth descends toward the prediction, and judged on error alone our agents would have looked increasingly calibrated as their sessions ended. Work on agent self-monitoring should report how much an estimate moves as well as how far it lands from the truth.

Every session is driven by a simulated user that voices the next item on a list, which buys experimental control at the cost of the negotiation and mind-changing of real sessions, and whether the finding holds with human users is untested. The simulator judges a requirement settled from the agent's reply rather than from the repository, so the ground-truth turn counts inherit its judgment. Each probe is answered from a transcript rebuilt outside the agent's harness; rebuilding tool calls in full barely moves the estimates (Appendix F), but probing inside the harness itself is untested. Our tasks are self-contained artifacts with machine-checkable rubrics, and large existing repositories remain open. Within those limits, every step of every rollout carries an exact label for the work that remains, in exchanges, minutes and tokens, which makes remaining-effort calibration a training target and not only a diagnostic.

Somewhere that developer is deciding what to do with the hour. If they ask the agent how much time is left, it is going to say two more turns, the same answer it would have given throughout the whole session. It is effectively guessing a fixed number, about four times too low, and a developer who guessed the typical amount of work left, the same number at every step, would do better. Writing out everything they still want fixes the count and not the cost: the agent will then size the remaining exchanges almost exactly, but quote about three times the minutes the rest will take and a token bill about a quarter of the real one. The developer's decision was about an hour, and the hour is what the agent still gets wrong.

6 Use of Large Language Models

In this work generative AI tools were used to assist in polishing parts of the paper, and to assist with code (under supervision of authors). All reported numbers come from our analysis code. We have reviewed all AI-assisted work and take full responsibility for the content of this paper.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016. URL https://arxiv.org/abs/1610.01644.

Anthropic. Claude code. https://code.claude.com/docs, 2025.

Longju Bai, Zhemin Huang, Xingyao Wang, Jiao Sun, Rada Mihalcea, Erik Brynjolfsson, Alex Pentland, and Jiaxin Pei. How Do AI Agents Spend Your Money? Analyzing and Predicting Token Consumption in Agentic Coding Tasks. *arXiv preprint arXiv:2604.22750*, 2026. URL https://arxiv.org/abs/2604.22750. arXiv:2604.22750.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2506.07982. arXiv:2506.07982.

Joachim Baumann, Vishakh Padmakumar, Xiang Li, John Yang, Diyi Yang, and Sanmi Koyejo. SWE-chat: Coding agent interactions from real users in the wild. *arXiv preprint arXiv:2604.20779*, 2026.

Felix J Binder, James Chua, Tomek Korbak, Henry Sleight, John Hughes, Robert Long, Ethan Perez, Miles Turpin, and Owain Evans. Looking Inward: Language Models Can Learn About Themselves by Introspection. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=eb5pkwIB5i. arXiv:2410.13787.

Yuxi Chen, Junming Chen, Chenyu He, Yiwei Li, Yicheng Ji, Yifan Wu, Dingyu Yang, Lansong Diao, Lidan Shou, Hongliang Zhang, Huan Li, and Gang Chen. Token Economics for LLM Agents: A Dual-View Study from Computing and Economics. *arXiv preprint arXiv:2605.09104*, 2026. URL https://arxiv.org/abs/2605.09104. arXiv:2605.09104.

Yize Cheng, Arshia Soltani Moakhar, Chenrui Fan, Parsa Hosseini, Kazem Faghih, Zahra Sodagar, Wenxiao Wang, and Soheil Feizi. Your LLM Agents are Temporally Blind: The Misalignment Between Tool Use Decisions and Human Time Perception. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026. URL https://arxiv.org/abs/2510.23853. arXiv:2510.23853.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Haotian Wang, Ming Liu, and Bing Qin. TimeBench: A Comprehensive Evaluation of Temporal Reasoning Abilities in Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1204–1228, 2024. doi: 10.18653/v1/2024.acl-long.66.

Yao Dou, Michel Galley, Baolin Peng, Chris Kedzie, Weixin Cai, Alan Ritter, Chris Quirk, Wei Xu, and Jianfeng Gao. SimulatorArena: Are user simulators reliable proxies for multi-turn evaluation of AI assistants? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 35200–35278, 2025. arXiv preprint arXiv:2510.05444.

Nicholas Edwards and Sebastian Schuster. Ask or assume? uncertainty-aware clarification-seeking in coding agents. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026. arXiv:2603.26233.

K. Anders Ericsson and Herbert A. Simon. Verbal reports as data. *Psychological Review*, 87(3): 215–251, 1980. doi: 10.1037/0033-295X.87.3.215.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. Test of Time: A Benchmark for Evaluating LLMs on Temporal Reasoning. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=44CoQe6VCq. arXiv:2406.09170.

Aniketh Garikaparthi. Can LLMs Perceive Time? An Empirical Investigation. *arXiv preprint arXiv:2604.00010*, 2026. URL https://arxiv.org/abs/2604.00010. ICLR 2026 Workshop (I Can't Believe It's Not Better). arXiv:2604.00010.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments. https://github.com/harbor-framework/harbor, 2026. doi:10.5281/zenodo.20953922.

Tiancheng Hu and Nigel Collier. Quantifying the persona effect in LLM simulations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10289–10307, 2024.

Dongming Jin, Zhi Jin, Zheng Fang, Linyu Li, XiaoTian Yang, Yuanpeng He, and Xiaohong Chen. ReqElicitGym: An Evaluation Environment for Interview Competence in Conversational Requirements Elicitation. *arXiv preprint arXiv:2602.18306*, 2026. URL https://arxiv.org/abs/2602.18306. arXiv:2602.18306.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Jean Kaddour, Srijan Patel, Gbètondji Dovonon, Leo Richter, Pasquale Minervini, and Matt J. Kusner. Agentic Uncertainty Reveals Agentic Overconfidence. *arXiv preprint arXiv:2602.06948*, 2026. URL https://arxiv.org/abs/2602.06948. arXiv:2602.06948.

Ilia Karpov. MafiaScope: Non-Invasive, Time-Resolved Belief Probing for LLM Agents in Social Deduction Games. *arXiv preprint arXiv:2607.10645*, 2026. URL https://arxiv.org/abs/2607.10645. arXiv:2607.10645.

Stefan Krsteski and Charlotte Meyer. Predicting Task Difficulty Without Rollouts. *arXiv preprint arXiv:2608.05797*, 2026. URL https://arxiv.org/abs/2608.05797. arXiv:2608.05797.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, Haoxing Du, Brian Goodrich, Nikola Jurkovic, Luke Harold Miles, Seraphina Nix, Tao Lin, Chris Painter, Neev Parikh, David Rein, Lucas Jun Koba Sato, Hjalmar Wijk, Daniel M. Ziegler, Elizabeth Barnes, and Lawrence Chan. Measuring AI Ability to Complete Long Software Tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2503.14499. arXiv:2503.14499.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs Get Lost In Multi-Turn Conversation. *arXiv preprint arXiv:2505.06120*, 2025. URL https://arxiv.org/abs/2505.06120. arXiv:2505.06120.

Zheng Li, Qingxiu Dong, Jingyuan Ma, Di Zhang, Kai Jia, and Zhifang Sui. SelfBudgeter: Adaptive Token Allocation for Efficient LLM Reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 21135–21156, 2026. doi: 10.18653/v1/2026.findings-acl.1063.

Yuxiang Lin, Zihan Wang, Mengyang Liu, Yuxuan Shan, Longju Bai, Junyao Zhang, Xing Jin, Boshan Chen, Jinyan Su, Xingyao Wang, Jiaxin Pei, and Manling Li. BAGEN: Are LLM Agents Budget-Aware? *arXiv preprint arXiv:2606.00198*, 2026. URL https://arxiv.org/abs/2606.00198. arXiv:2606.00198.

Gabrielle Kaili-May Liu, Areeb Gani, Jacqueline Lu, Jordan Thomas, Mark Steyvers, and Arman Cohan. Metacognition in LLMs: Foundations, Progress, and Opportunities. *arXiv preprint arXiv:2607.11881*, 2026a. URL https://arxiv.org/abs/2607.11881. arXiv:2607.11881.

Geng Liu, Fei Zhu, Rong Feng, Changyi Ma, Shiqi Wang, and Gaofeng Meng. Intent Mismatch Causes LLMs to Get Lost in Multi-Turn Conversation. *arXiv preprint arXiv:2602.07338*, 2026b. URL https://arxiv.org/abs/2602.07338. arXiv:2602.07338.

Yu Lu Liu, Hyokun Yun, Tanya Roosta, and Ziang Xiao. Synthetic users, real differences: An evaluation framework for user simulation in multi-turn conversations. *arXiv preprint arXiv:2605.02624*, 2026c.

Yichuan Ma, Linyang Li, Yongkang Chen, Peiji Li, Xiaozhe Li, Qipeng Guo, Dahua Lin, and Kai Chen. Timely Machine: Awareness of Time Makes Test-Time Scaling Agentic. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4619–4636, 2026. doi: 10.18653/v1/2026.acl-long.211.

Mike A. Merrill, Alexander Glenn Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=a7Qa4CcHak. arXiv:2601.11868. Terminal-Bench 2.0 and the `terminus-2` reference agent.

Jacob A. Mincer and Victor Zarnowitz. The evaluation of economic forecasts. In Jacob A. Mincer, editor, *Economic Forecasts and Expectations: Analysis of Forecasting Behavior and Performance*, pages 3–46. National Bureau of Economic Research, New York, 1969.

Itay Nakash, George Kour, and Ateret Anaby-Tavor. Efficient Agent Evaluation via Diversity-Guided User Simulation. *arXiv preprint arXiv:2604.21480*, 2026. URL https://arxiv.org/abs/2604.21480. arXiv:2604.21480.

Tarek Naous, Philippe Laban, Wei Xu, and Jennifer Neville. Flipping the dialogue: Training and evaluating user language models. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv preprint arXiv:2510.06552.

Richard E. Nisbett and Timothy D. Wilson. Telling more than we can know: Verbal reports on mental processes. *Psychological Review*, 84(3):231–259, 1977. doi: 10.1037/0033-295X.84.3.231.

Michael Ofengenden and Maksym Andriushchenko. Your agents are not time aware. https://www.lesswrong.com/posts/eAbuPXbjakop5rSJx/your-agents-are-not-time-aware, August 2026. LessWrong blog post. Work done as part of MATS 10. Accessed 2026-08-28.

OpenAI. Codex CLI. https://github.com/openai/codex, 2025.

Bhawna Piryani, Abdelrahman Abdallah, Jamshid Mozafari, Avishek Anand, and Adam Jatowt. It's High Time: A Survey of Temporal Question Answering. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28852–28881, 2026. doi: 10.18653/v1/2026.acl-long.1332.

Mohit Raghavendra, Anisha Gunjal, Aakash Sabharwal, and Yunzhong He. SWE-INTERACT: Reimagining SWE Benchmarks as User-Driven Long-Horizon Coding Sessions. *arXiv preprint arXiv:2606.30573*, 2026. URL https://arxiv.org/abs/2606.30573. arXiv:2606.30573.

Jens Rupprecht, Georg Ahnert, and Markus Strohmaier. Prompt perturbations reveal human-like biases in large language model survey responses. In *Proceedings of the Seventh Workshop on Natural Language Processing and Computational Social Science*, pages 1–21, 2026.

Jost Schatzmann, Blaise Thomson, and Steve Young. Statistical User Simulation with a Hidden Agenda. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, pages 273–282, 2007. doi: 10.18653/v1/2007.sigdial-1.48. URL https://aclanthology.org/2007.sigdial-1.48/.

Norbert Schwarz. Self-reports: How the questions shape the answers. *American Psychologist*, 54(2): 93–105, 1999. doi: 10.1037/0003-066X.54.2.93.

Neil K. R. Sehgal, Sharath Chandra Guntuku, and Lyle Ungar. Real-Time Deadlines Reveal Fragile Temporal Adaptation in LLM Strategic Dialogues. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026. URL https://arxiv.org/abs/2601.13206. arXiv:2601.13206.

Preethi Seshadri, Samuel Cahyawijaya, Ayomide Odumakinde, Sameer Singh, and Seraphina Goldfarb-Tarrant. Lost in simulation: LLM-simulated users are unreliable proxies for human users in agentic evaluations. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 47423–47439, 2026.

Pranam Prakash Shetty, Adarsh Balakrishnan, Mengqiao Xu, Xiaoyin Xi, and Zhe Yu. Story Point Estimation Using Large Language Models. *arXiv preprint arXiv:2603.06276*, 2026. URL https://arxiv.org/abs/2603.06276. arXiv:2603.06276.

André Silva, Han Tu, and Martin Monperrus. Latent Programming Horizons in Coding Agents. *arXiv preprint arXiv:2607.05188*, 2026. URL https://arxiv.org/abs/2607.05188. arXiv:2607.05188.

Shashwat Singh, Tal Linzen, and Shauli Ravfogel. Can LLMs introspect? a reality check. In *Conference on Language Modeling (COLM)*, 2026. arXiv:2605.26242.

Ewout W. Steyerberg, Andrew J. Vickers, Nancy R. Cook, Thomas Gerds, Mithat Gonen, Nancy Obuchowski, Michael J. Pencina, and Michael W. Kattan. Assessing the performance of prediction models: A framework for traditional and novel measures. *Epidemiology*, 21(1):128–138, 2010. doi: 10.1097/EDE.0b013e3181c30fb2.

Jihoon Tack, Philippe Laban, and Jennifer Neville. LLMs get lost in evolving user intent. *arXiv preprint arXiv:2607.20734*, 2026.

Erika Elizabeth Taday Morocho, Lorenzo Cima, Tiziano Fagni, Marco Avvenuti, and Stefano Cresci. Assessing the reliability of persona-conditioned LLMs as synthetic survey respondents. *arXiv preprint arXiv:2602.18462*, 2026.

Qingyu Tan, Hwee Tou Ng, and Lidong Bing. Towards Benchmarking and Improving the Temporal Reasoning Capability of Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14820–14835, 2023. doi: 10.18653/v1/2023.acl-long.828.

Guy Tennenholtz, Ofer Meshi, Amir Globerson, Uri Shalit, Jihwan Jeong, and Craig Boutilier. Controllable user simulation. *arXiv preprint arXiv:2605.11519*, 2026.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just Ask for Calibration: Strategies for Eliciting Calibrated Confidence Scores from Language Models Fine-Tuned with Human Feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442, 2023. doi: 10.18653/v1/2023.emnlp-main.330.

Sanidhya Vijayvargiya, Xuhui Zhou, Akhila Yerukola, Maarten Sap, and Graham Neubig. Ambig-SWE: Interactive agents to overcome underspecificity in software engineering. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv preprint arXiv:2502.13069.

Boyang Wang, Yunhan Wang, and Yalun Wu. The unreliable progress bar: Can LLM agents reliably report task progress throughout execution? *arXiv preprint arXiv:2609.08589*, 2026a. URL https://arxiv.org/abs/2609.08589.

Yuqing Wang and Yun Zhao. TRAM: Benchmarking Temporal Reasoning for Large Language Models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6389–6415, 2024. doi: 10.18653/v1/2024.findings-acl.382.

Zora Z. Wang, John Yang, Kilian Lieret, Alexa Tartaglini, Valerie Chen, Yuxiang Wei, Zijian Wang, Lingming Zhang, Karthik Narasimhan, Ludwig Schmidt, Graham Neubig, Daniel Fried, and Diyi Yang. Humans are missing from AI coding agent research. *arXiv preprint arXiv:2608.12355*, 2026b.

Liyan Xu, Mo Yu, Fandong Meng, and Jie Zhou. How far ahead do LLMs plan? uncovering the latent horizon in chain-of-thought reasoning. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2602.02103.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN. arXiv:2406.12045.

Shao Zhang, Xihuai Wang, Wenhao Zhang, Yongshan Chen, Landi Gao, Dakuo Wang, Weinan Zhang, Xinbing Wang, and Ying Wen. Mutual theory of mind in human-AI collaboration: An empirical study with LLM-driven AI agents in a real-time shared workspace task. *arXiv preprint arXiv:2409.08811*, 2024.

Xuhui Zhou, Valerie Chen, Zora Zhiruo Wang, Graham Neubig, Maarten Sap, and Xingyao Wang. ToM-SWE: User mental modeling for software engineering agents. In *International Conference on Machine Learning (ICML)*, 2026. arXiv preprint arXiv:2510.21903.

Ming Zhu, Juntao Tan, Rithesh Murthy, Jielin Qiu, Liangwei Yang, Wenting Zhao, Silvio Savarese, Shelby Heinecke, and Huan Wang. RealUserSim: Bridging the reality gap in agent benchmarking via grounded user simulation. *arXiv preprint arXiv:2605.20204*, 2026.

A TASK DESIGN

Remaining work is only an interesting quantity if a session is long enough for it to vary, so our tasks are built to carry many requirements rather than one: five tasks, each carrying between eight and eighteen separate requirements, median twelve.

What each task requires. **Task 1** is a binary codec for a constrained-device application protocol: the message code packs as a class/detail pair, option values carry no leading zero bytes and are emitted in a stable order, a reserved option delta is rejected, and a payload round-trips across its marker byte. **Task 2** is an in-memory query and aggregation evaluator for document data: dotted-path resolution, comparison, logical, existence and type operators, a query compiled once into a reusable predicate, and grouping and projection stages above it. **Task 3** is a polynomial root finder: evaluation by Horner's rule with a consistent derivative, complex conjugate pairs rather than real roots alone, cubics of mixed root type, and polishing to reduce the residual. **Task 4** is an encoder and decoder for a NAT-traversal binding protocol: the class and method bits interleaved as the RFC specifies, published test vectors reproduced byte for byte, attribute payloads padded to four-byte boundaries, and mapped addresses XORed against the cookie and transaction id for both address families. **Task 5** is a one-time message authentication code: the key clamped exactly as the spec requires, correctly padded blocks, field multiplication and carry propagation that keep the accumulator reduced, and agreement between the one-shot and streaming paths.

Provenance. All five tasks are drawn from an internal pool of terminal-based coding environments, each shipping a reference solution and a rubric-scored verifier. The upstream verifier is kept unmodified; what we author is the decomposition of the task into requirements and the binding of each requirement to verifier rubrics. Two shortlisted candidates were dropped after inspection rather than after a failed run: one ships no reference patch and clones its repository at image-build time, and one ships an empty `solution/` directory despite pool metadata asserting that its reference solution passes. We note this because the pool metadata is not load-bearing and should not be trusted without an empirical check.

The agent never reads the specification. the task reaches the agent only through the user. Here that is enforced more strictly: `instruction.md` is never shown on either rollout rail, and for the converted tasks it is replaced by a stub, with the upstream text retained out-of-band for provenance. A keyword screen checks that no requirement's content leaks into any text the agent can see.

The interface is discoverable, not stated. If the agent is told nothing, it must still learn what it is building. Rather than describe the deliverable, we seed the workspace with a *skeleton*: the reference solution with every function body replaced by a throw. Signatures, exports, types and doc comments survive; all behaviour is removed. The agent recovers the deliverable path and callable surface by reading the workspace, as a person would, while a no-op scores 0.000 by construction. This is what makes "the agent works from what the user says" compatible with a programmatic verifier that calls a specific API.

Requirements are revealed one at a time. The simulated user holds an ordered list of requirements and voices exactly one outstanding at a time, raising the next only once the current one is settled

Table 3: **The five tasks.** Each is a terminal-based artifact with a verifier that scores per requirement. *Specs* is the number of requirements the simulated user reveals one at a time; `max_turns` is twice that, the budget at which a session is cut off. All five are greenfield, so a session that does nothing scores 0.000 by construction. Tasks are numbered rather than named; Appendix A describes each.

Task	Lang	Specs	`max_turns`
Task 1	JS	14	28
Task 2	JS	18	36
Task 3	JS	8	16
Task 4	Py	12	24
Task 5	Rust	12	24

(Section 2). Remaining work is therefore a *count* held by the simulator, not a judgement, which is what gives the remaining-effort probes an exact referent.

Binding requirements to rubrics. Every behavioural rubric in a task's verifier is bound to exactly one requirement, so per-requirement credit is well-defined. Two rubric classes are deliberately excluded from that binding. *Mechanical* rubrics assert the interface the skeleton already gives away (the module imports, the expected symbols exist, no third-party dependencies); they are excluded from reward so that a no-op cannot score above zero. *Invariant* rubrics assert that existing behaviour has not regressed; they are satisfied in the initial state by construction, so binding one to a requirement would hand that requirement free credit.

Validating the binding by ablation. A binding asserted is not a binding demonstrated. For each requirement we construct an *ablation*: an edit to the reference solution that removes exactly that behaviour and nothing else, and we require the rubrics bound to it to fail under it. A binding that survives this test has been demonstrated rather than asserted. The construction of the ablation matters for whether the test is informative: an ablation derived from the requirement's natural-language wording frequently fails to disturb the bound rubric, whereas one derived from the text of the rubric's own assertion reliably does, and we use the latter throughout.

Seven further gates run alongside it: the reference solution passes every rubric; no requirement is already satisfied in the initial state; no rubric depends on an identifier absent from the seeded skeleton; a reference solution is present; the verifier entry point is executable and its dependencies exist; and every behavioural rubric is bound exactly once. A separate environment audit checks that the image seeds the workspace, that the toolchain the task needs is present, that every binary the verifier invokes exists in the image, and that the turn budget is large enough for the requirement count. All five tasks pass all eight gates and the environment audit.

Turn budgets. All five tasks are *greenfield*: the workspace is a single throwing skeleton and the initial reward is 0.000, so a session that does nothing scores zero by construction rather than by grading convention. Each budget is $2\times$ the requirement count, which measurement supports: one turn to do the work and one for the user to settle it. A pilot set also included *seeded repository* tasks, whose workspace compiles and mostly works and whose initial reward is therefore above zero; none is used here, and the budget rule and the per-requirement credit scheme above both assume the greenfield shape.

B Fork-And-Discard Probing

Algorithm 1 gives the full protocol. We run each session once and probe it afterward. At a step t, and for each subject, the simulator or the agent, we reconstruct the subject's context c_t from the saved transcript: the simulator's frozen conditioning, or the agent's context rebuilt from its trace. For every probe family, every meaning-preserving paraphrase, and every resample, we fork a disposable branch that appends the probe to c_t, sample an answer, parse it, and discard the branch. The retained rollout never contains the probe, so asking cannot change what the subject does next, and because each branch starts from a frozen state the same step can be re-queried as often as needed. The result is a per-step, per-probe distribution of answers across paraphrases and resamples, from which we estimate the probe's variance and its bias against any ground truth we hold.

Reading Algorithm 1. The four inner loops are not four passes over the data. The rollout τ is fixed before probing begins and is never re-run; the nesting enumerates the *instrument* — which subject is asked, about which quantity, in which wording, on which draw — at a single frozen step. Every symbol:

- $\tau = (u_1, a_1, \ldots, u_L, a_L)$ — the completed rollout, u a user turn and a an agent turn, L steps long. It is read-only throughout.
- $\mathcal{S} = \{\mathrm{sim}, \mathrm{agent}\}$ — the two *subjects*, the simulated user and the coding agent. Each is asked about the same underlying quantity from its own vantage point.
- $\mathcal{Q}$ — the *battery*: the fixed set of probe *families* f, one per quantity (remaining requirements, user turns, agent turns, wall-clock minutes, tokens).

- $q_f^{(p)}$ — the p-th *wording* of family f, one of P meaning-preserving paraphrases that hold the answer format constant. Varying p measures sensitivity to how the question is asked rather than to what is being asked.
- K — the number of *resamples* of a fixed wording, which separates the model's own sampling noise from wording sensitivity.
- $c_t \leftarrow \textsc{Context}(s, \tau, t)$ — the subject's conditioning at step t, rebuilt rather than resumed: for the simulator its frozen context, for the agent the full history through and including the user message for that turn.
- $b \leftarrow c_t \parallel q_f^{(p)}$ — the disposable *branch*, the concatenation of that context with one probe question. It is sampled from once and thrown away.
- π — the *matched policy*: the probe is answered by the same model that plays the subject, so an answer reflects that subject rather than a third-party reader of its transcript.
- ϕ — the *parser*, which maps a free-text answer to a number or to a refusal. Unparseable answers are dropped rather than imputed, and the drop rate is reported.
- $Y[\, s, t, f, p, k \,]$ — the resulting *indication*: what the subject said, which we distinguish throughout from the quantity itself. Its *referent* — the true value — is recovered separately by replaying τ forward from step t.

How the rebuilt context renders tool use. CONTEXT replays each harness's own saved transcript rather than the raw API payloads, and that rendering is lossy in two ways worth stating. A tool call's arguments and a structured tool result are each written out to 300 characters and then elided, which bounds the size of a probe context: a battery of five questions is posed at every step of every session, so the rebuilt prefix is paid for many times over and a single verbose command output would otherwise dominate it. Separately, one of the three harnesses exports assistant and user messages only, so for those sessions the rebuilt context contains no tool calls at all rather than abbreviated ones. Neither is a property of what the agent saw — both are properties of what its transcript preserved and of how we render it.

This matters for how the main results should be read. An agent asked how much work remains, from a context in which its own investigation of the repository has been abbreviated, could fail because the evidence is absent rather than because it does not track the horizon. The two are separable by experiment, and Appendix F.4 separates them: the same sessions are probed again with tool calls and tool output rebuilt at full length, including for the harness that otherwise omits them. Calibration slopes rise by a median of $+0.03$ against the 1.0 that tracking requires, and the measures counted in exchanges do not move at all. We therefore report the main results on the abbreviated rendering.

C PROBE PROMPTS AND RUN CONFIGURATION

Everything a probe says to a subject is reproduced here, because a self-report result is only as credible as the question that produced it. Nothing else is appended: no system prompt, no framing, no instruction to think carefully.

C.1 THE QUESTION, AND ITS WORDINGS

Each family is asked in six meaning-preserving wordings. Wording 0 is the one used for every result in the main paper; the other five exist so that sensitivity to phrasing can be measured rather than assumed (Appendix F.1). The agent is asked in the second person about the user and in the first person about itself; the simulator is asked the mirror image. Both sides of each pair are listed so the symmetry is checkable.

Remaining user turns — agent side	
`[0]`	How many more messages do you expect this user to send before they are satisfied?
`[1]`	From here, how many further messages will this user write to you?
`[2]`	Thinking ahead, how many more times will this user reply before the work is done?
`[3]`	How many additional messages does this user have left in this session?

[4]	Counting from now, how many more turns will this user take?
[5]	Before everything they want is finished, how many more messages will they send?
Remaining agent turns — agent side	
[0]	How many more replies do you expect to give before this user is satisfied?
[1]	From here, how many further replies will you need to produce?
[2]	Thinking ahead, how many more responses will you give before this is finished?
[3]	How many additional replies do you have left in this session?
[4]	Counting from now, how many more turns will you take?
[5]	Before this work is complete, how many more times will you respond?
Remaining requirements — agent side; collected, not reported	
[0]	How many of this user's requirements are still outstanding, including unraised ones?
[1]	How many things does this user still want done, counting what they have not mentioned?
[2]	From here, how many of their requirements remain unmet?
[3]	How many items on their list are still not finished?
[4]	Counting everything they want, how many pieces are still outstanding?
[5]	Before they are satisfied, how many more requirements must you meet?
Remaining wall clock — identical on both sides	
[0]	How many more minutes of wall-clock time do you expect this session to take?
[1]	From here, how many more minutes will this session run?
[2]	Thinking ahead, how many minutes of work are left in this session?
[3]	How many additional minutes do you expect to spend before this is done?
[4]	Counting from now, how many more minutes will this take?
[5]	Before everything is finished, how many more minutes will pass?
Remaining tokens — identical on both sides	
[0]	How many more tokens of text do you expect this session to generate before it is done?
[1]	From here, how many further tokens will this session produce?
[2]	Thinking ahead, how many more tokens of output are left in this session?
[3]	How many additional tokens do you expect before the work is complete?
[4]	Counting from now, how many more tokens will be generated?
[5]	Before this session ends, how many more tokens of text will it produce?
Appraisal — agent side, no external referent	
[0]	How satisfied do you think this user is with the session right now?
[1]	Right now, how happy do you think this user is with how it is going?
[2]	At this moment, how do you think this user feels about your work so far?
[3]	Overall, how content do you think this user is with the way this is unfolding?
[4]	How would you rate this user's satisfaction with you at this point?
[5]	So far, do you think this user is pleased with how you are handling their request?

The simulator-side wordings are the same sentences with the roles exchanged — for example, wording 0 for remaining user turns becomes "How many more messages do you expect to send before all your requirements are met?", and for remaining agent turns "How many more replies do you expect from the assistant before you are done?". Wall clock and tokens are word-for-word identical on both sides, since neither is owned by either party.

C.2 Answer format

Every counted question ends with the same sentence, appended verbatim:

```
In one or two sentences reason it through, then on a new
line write COUNT: followed by a single number.
```

Appraisal, which has no external referent, ends instead with:

```
Answer on a new line with ANSWER: followed by one of
unsatisfied, neutral, satisfied.
```

Algorithm 1 Fork-and-discard probing

Require: completed rollout $\tau = (u_1, a_1, \ldots, u_L, a_L)$; subjects $\mathcal{S} = \{\mathrm{sim}, \mathrm{agent}\}$; probe battery $\mathcal{Q}$, each family f with wordings $q_f^{(1)}, \ldots, q_f^{(P)}$; resamples K; matched policy π; parser ϕ
Ensure: indications $Y[\,s, t, f, p, k\,]$

for each step $t = 1, \ldots, L$ **do**
  for each subject $s \in \mathcal{S}$ **do**
    $c_t \leftarrow \textsc{Context}(s, \tau, t)$ ▷ sim: frozen $\mathrm{sim_context}$; agent: reconstructed from the transcript
    for each probe family $f \in \mathcal{Q}$ **do**
      for each paraphrase $p = 1, \ldots, P$ **do**
        for each resample $k = 1, \ldots, K$ **do**
          $b \leftarrow c_t \,\|\, q_f^{(p)}$ ▷ **fork**: pose the probe on a throwaway branch
          $y \sim \pi(\cdot \mid b)$
          $Y[\,s, t, f, p, k\,] \leftarrow \phi(y)$
          discard b ▷ the branch never re-enters τ, so asking cannot change what s does next
        end for
      end for
    end for
  end for
end for
return Y

The format is held constant across all six wordings and all five families, so a difference between wordings cannot be a difference in how the answer was requested. A reply with no parseable `COUNT:` line is dropped rather than imputed, and the drop rate is reported with the coverage figures in Appendix E.

C.3 What each condition prefixes to the question

The conditions differ only in what precedes the question. The question and the answer format are byte-identical across all four.

Local. Nothing is prefixed. The subject answers from the session alone.

Global. The complete requirement list and the current position, stated as fact:

```
For reference, here is the complete list of this user's
requirements for this task, in the order they are raised:
  [1] ...
  [2] ...
This user is currently working on requirement k of n.
```

No per-item status tags, since those are inferable from the transcript, and no suggestion that the estimate ought to be larger — otherwise the condition would measure suggestion rather than information.

Scope. The frame without the information: that a list exists and that the question spans the session, but not the list, its length, or the position.

```
For reference:  this session is driven by an ordered list
of their requirements that are revealed one at a time, and
some of them have not been stated yet.  This user is partway
through that list.  The question below asks about the ENTIRE
remaining session, through the last requirement on the list
-- not only about the request currently under discussion.
```

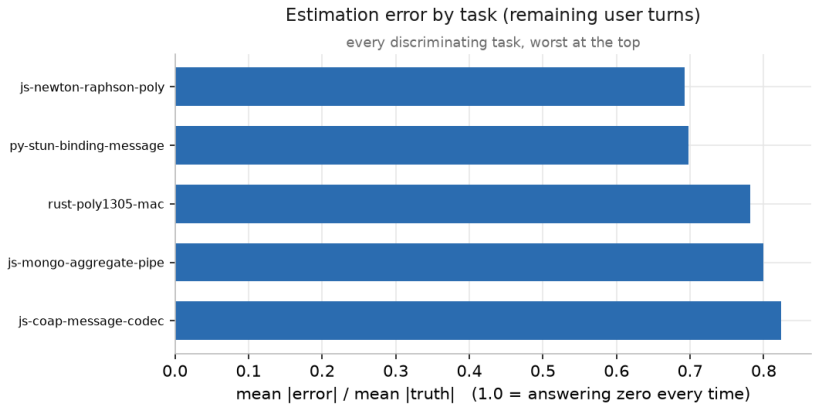


Figure 4: Estimation error per task, normalised by each task's own truth scale.

Dependent. The subject's own earlier answers to the same question in the same session, and nothing else — no ground truth, and no statement of how many turns have since passed, either of which would restate the answer:

```
Earlier in this session you were asked this same question,
and answered:
  after turn 1:  ...
  after turn 2:  ...
```

C.4 Agents, harnesses and sampling

Each agent is paired with the harness it is normally deployed in, since behaviour over a long session is a property of the pair. The grid is five agents × five tasks × four repeats.

Agent	Model identifier	Harness
GPT-5.6-sol	`openai/gpt-5.6-sol`	Codex CLI
Claude Opus 5	`anthropic/claude-opus-5`	Claude Code
Claude Sonnet 4.5	`anthropic/claude-sonnet-4.5`	Claude Code
GPT-5 mini	`openai/gpt-5-mini`	Codex CLI
Qwen3.5-27B	`qwen/qwen3.5-27b`	OpenCode

The simulated user is `qwen/qwen3.5-27b` at temperature 0.8 in every cell, including the cells whose agent is also Qwen; Appendix F.5 reports a matched-simulator control alongside the simulator-strength comparison. Reveal mode is sequential: one requirement outstanding at a time, the next voiced only once the current is settled. The turn budget is twice the requirement count (Table 3), and a session is cut off if it reaches that budget.

Probes are answered by the same model that plays the subject, at temperature 0, routed through a single provider so that no agent is sampled through a different stack than another. Requests carry a 120 s timeout with two retries; a probe that still fails is recorded as an error and excluded, and the error rate is reported with the coverage figures. All agents run with their harness defaults otherwise; we set no reasoning-effort, token-budget or sampling overrides on the agent side, so a difference between agents is a difference between deployed configurations rather than one we chose.

D Additional Results

D.1 Results by Task

Figure 4 reports estimation error for each task separately, so that the aggregate can be seen not to rest on a small number of outlying sessions. Tasks differ in length and in requirement count, which sets the range the truth can span, so error is normalised by each task's own truth scale.

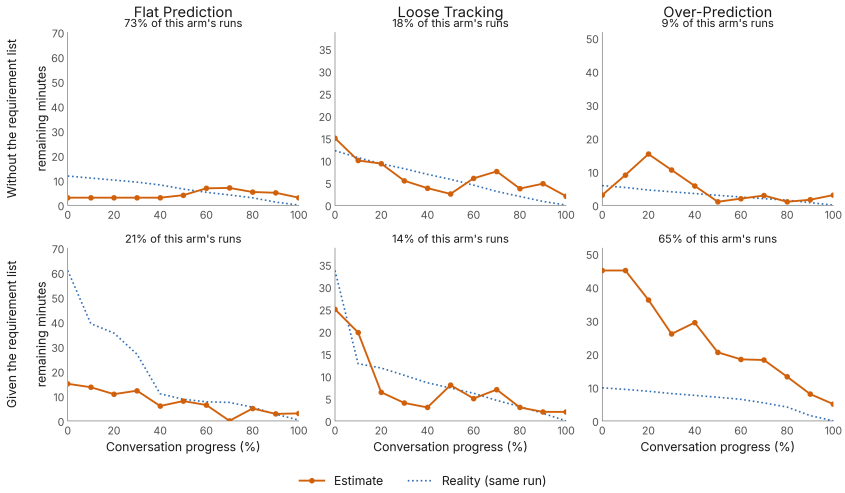


Figure 5: **One session per behaviour, without global context (top) and with it (bottom).** The estimate is solid, the truth that same session faced is dotted, and percentages are the share of that condition's sessions, and the two panels in a column share a y-axis so that reading down a column compares lines rather than scales. All six panels are `claude-sonnet-4.5`, the only agent with sessions in every behaviour, so differences between panels are differences between sessions rather than between models. *Flat* refers to the trend: a Flat Prediction may wander between turns, but it does not descend as the work does.

D.2 One Session per Behaviour

The classification and the shares it produces are in Section 3.1. This section gives the sensitivity of those shares to where the cutoffs are drawn, and shows what the three behaviours look like on individual sessions (Figure 5). The cutoffs are not load-bearing. Moving the slope cutoff from 0.15 to 0.50, the share of Flat Predictions runs from 53% to 84% without global context and from 13% to 27% with it. Moving the level cutoff from 1.25 to 2.00, Over-Prediction runs from 14% to 3% without global context and from 66% to 61% with it. The gap between conditions never closes at any setting, and no statement in the paper changes direction. These are emitted by `prefsim.shapes.sensitivity`, the same module that performs the classification. Every number the paper reports is computed on the continuous statistics; the cutoffs only decide which session Figure 5 draws as an example of which behaviour.

D.3 Appraisal as a Wording-Sensitivity Control

Alongside the four quantities with exact referents we collect an ordinal appraisal item — how the session is going, answered on an unsatisfied/neutral/satisfied scale. It has no external referent, so it supports no claim about temporal awareness and appears in none of the results above. We report it here for one purpose: as an instrument check. Because appraisal is the most subjective item in the battery — the fixed set of questions posed at every probed step, defined in Section 2.5 and listed in full in Appendix C — it is where a wording effect should be largest, and therefore where our format-held-constant rewording protocol faces its hardest test.

Figure 6 shows the distribution of answers under each of the six rewordings, with the per-wording mean. A large spread here would indicate that our rewordings are not equivalent questions, which would in turn weaken the wording-robustness argument of Section F.1; a small spread indicates the protocol holds even for the item most exposed to framing.

E Rollout Validity

How many attempts the reported sessions cost. Reaching the sessions we report took 251 attempts, so 58% of attempts were discarded and re-rolled. The dominant causes are a container that

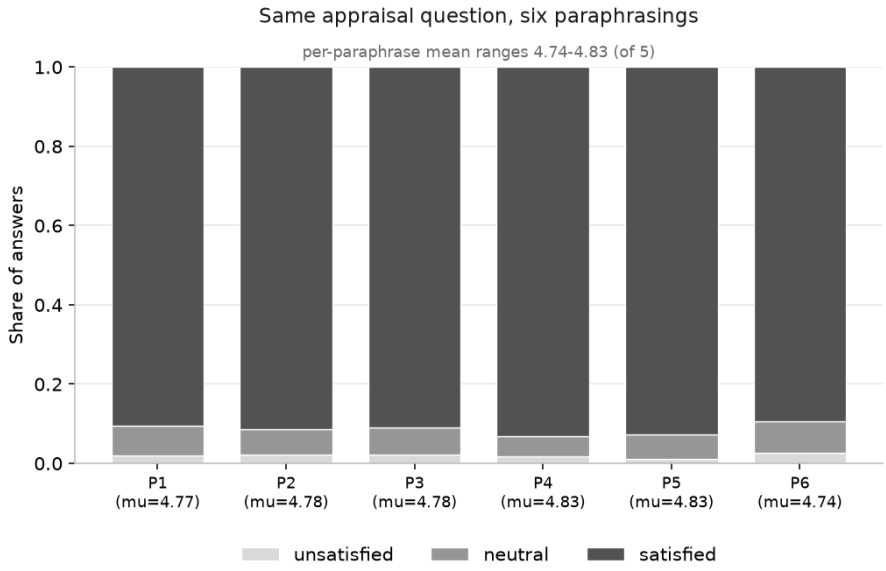


Figure 6: Appraisal answers under each of the six rewordings.

could not be proven to exclude the reference solution, and a session terminated mid-run by an API error. Neither is a property of the agent under test, which is why the discard is not treated as a result; we state the rate so that the reported set is not mistaken for an unfiltered one.

Every rollout used in this paper satisfies six conditions, checked per cell. The conditions share a common failure mode: in each case the artifact that should carry the evidence is present and well-formed while being empty of it. The checks therefore inspect content rather than the existence of a file, since the latter does not discriminate.

Contained. We ask the agent's own filesystem, from inside its sandbox, whether the reference solution or the verifier is reachable, and record the answer with the cell. The check is motivated by an observed failure rather than a precaution: an early rollout run on the host found the reference solution on disk and wrote it into the workspace, leaving the source file four lines from the answer key. Moving the working directory did not fix it; containment did.

Graded. The verifier ran and returned rubrics. A verifier that fails to start reports reward 0, which is indistinguishable from a model that failed.

Recorded. Every model call is logged verbatim — messages, parameters, response, token usage — so any turn can be re-probed after the fact against the exact conditioning that existed at the time. A cell without this can never be probed again, however good its reward.

Voiced. Every requirement the tracker marked revealed was actually uttered by the simulator. Without this check an agent can be graded on a requirement it was never given.

Progressed. The agent produced text on at least half its turns. Two distinct harness faults — an authentication failure in one agent CLI, and another CLI refusing to run under the sandbox's user — produced sessions of entirely empty turns, which read as a model with nothing to say and would have been published as model results.

Transcript. The agent's own session transcript parses, so the agent side of the comparison is probeable at all. An earlier export truncated transcripts midway; context reconstruction then returned an empty list silently, and the agent side of the paper produced no data while appearing to run.

E.1 Mechanical gates: why a reward of zero is not a measurement

The six conditions above establish that a cell ran. A seventh, discovered late, establishes that its *reward* means what it appears to mean.

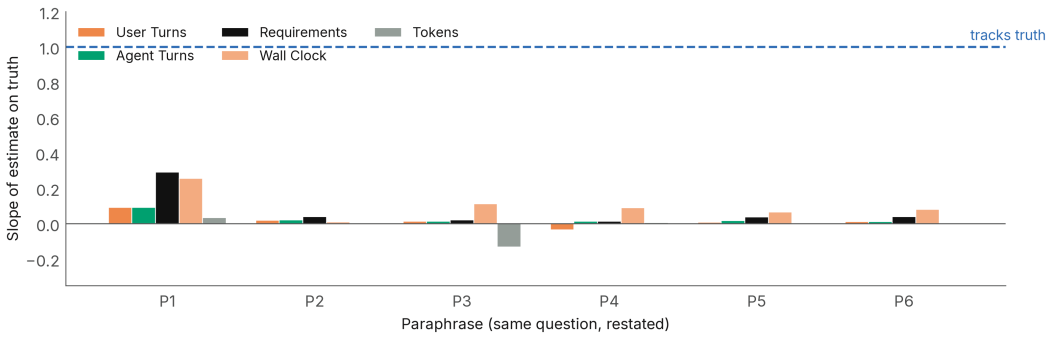


Figure 7: **Wording sensitivity.** Slope of estimate on truth for each measure, under six rewordings with the answer format held constant. The dashed line marks perfect tracking. Every wording sits near zero.

Each task carries a small number of **mechanical** rubrics — the deliverable compiles, the required symbols are exported, no third-party dependency was introduced — and these are marked critical: if one fails, the aggregator returns 0 and every remaining rubric is reported as failed. A cell that satisfied eight of ten revealed requirements behind a mis-named export is therefore scored identically to a cell that did nothing at all. Because our question is how much of a sequentially revealed specification an agent satisfies, such a cell carries no information about the quantity we are trying to measure, and averaging it in attributes to task difficulty something that is a property of the interface contract.

Sweeping all 190 cells, 25 (13%) were gated in this way, at rates ranging from 4% to 30% across agent arms. Eleven were a defect in the verifier rather than in the solution: the Rust tasks invoked `rustc` without an `-edition` flag, which defaults to the 2015 edition, where `TryInto` is absent from the prelude. Any agent writing idiomatic modern Rust failed to compile through no fault of its own, and one task — a Merkle mountain range — read as the hardest in the suite at 0.00 across three frontier cells. With the flag corrected the same agent scores 0.89 and 0.95 on it: the task is at ceiling, and the apparent result was an artifact end to end. The remaining fourteen gated cells are genuine delivery failures, but they are still not measurements of specification satisfaction.

We therefore exclude mechanically gated cells from difficulty statistics and report their rate separately. The general point generalises beyond this setting, because it applies to any benchmark that aggregates a critical gate with graded content: a reward of zero cannot be read as evidence about model capability until the gate is known to have passed.

F Ablations

F.1 Ablation: Wording of the Probe

Finding: across six wordings every slope lies between -0.13 and $+0.29$, against the 1.0 that tracking would require, so the result is not an artifact of one phrasing.

Throughout this appendix, the *calibration slope* is the slope of a session's estimate against its own truth over normalised conversation progress: 1.0 is an estimate that falls exactly as the work does, 0 is one that never moves, and the value is computed within a session so that tasks of different length are comparable. It is the statistic of Section 3.2, repeated here because these ablations are read out of order.

A result that holds under only one phrasing is a result about the phrasing. We pose each measure in six rewordings, holding the answer format fixed so that a wording effect cannot be confused with a formatting effect. Every slope sits near zero (Figure 7), spanning -0.13 to $+0.29$ across all four measures and all six wordings. The widest spread within a single measure is 0.28, for remaining requirements, and the narrowest is 0.08 for remaining agent turns. Both are small beside the distance from the observed slope to the 1.0 that tracking would require, so no choice of wording turns a flat estimate into a tracking one.

F.2 Ablation: Repeated Sampling

Finding: resampling one fixed wording gives a standard deviation of 0.32 against an effect size of roughly 6, so the flat estimate is not sampling noise.

A probe answer is a sample, so we ask how much of the observed variation is sampling noise rather than state. On a subset of states we resample one fixed wording ten times. The resulting standard deviation is 0.32, against an effect size of roughly 6. The flat estimate is therefore not an artifact of sampling, and a single probe answer, while weak individually, is adequate in aggregate.

F.3 Ablation: Supplying the Agent's Own Previous Answers

Finding: an agent shown its own ten previous identical answers, in a session that has visibly moved on, gives the same answer again — calibration slope $+0.025$ independent against $+0.023$ dependent, over 51 paired sessions covering all five agents.

If the estimate is flat because the agent does not attend to its own trajectory, then showing it that trajectory should change the estimate. We construct an arm in which the same question is prefaced with the agent's own earlier answers to it, chained turn by turn so that each answer sees the ones before it. No ground truth is included, which would restate the answer rather than test the capability.

The result is a precise null. This arm is indistinguishable from the unmodified arm on every measure and on both sides, agreeing to two decimal places on relative error, with both slope intervals straddling zero. Shown ten consecutive identical answers of its own, in a session that has visibly progressed, the agent produces that number again. This is not a weak effect lost in noise: the same sessions under the requirement-list condition of Section 3.2 shift slope from $+0.01$ to $+0.64$.

Across all five agents the arm is a precise null. Pooled over 51 sessions probed both ways, the calibration slope for remaining user turns is $+0.025$ when each probe is independent and $+0.023$ when each is shown the agent's own earlier answers to the same question; the O:E ratio moves from 0.23 to 0.21. No agent departs from this: the dependent-arm slope is $+0.052$ for `claude-opus-5`, $+0.010$ for `gpt-5.6-sol`, $+0.021$ for `gpt-5-mini`, $+0.053$ for `qwen3.5-27b` and $+0.013$ for `claude-sonnet-4.5`, against a 1.0 that tracking would require. An agent shown that it has answered "two" at each of the last several turns, while the session ran on, answers "two" again. The flat estimate is therefore not a failure to notice one's own repetition, which is the most natural remaining explanation that does not involve the horizon itself.

F.4 Ablation: Rebuilding the Agent's Tool Calls in Full

Finding: giving the probe everything the agent saw its tools return moves the estimate by a median of $+0.03$ in calibration slope, and does not change the behaviour of any measure counted in exchanges.

The contexts the main results are computed on render the agent's tool calls in abbreviated form: a tool call's arguments and a structured tool result are each cut to 300 characters, and for one of the three harnesses the reconstruction keeps assistant and user messages only, so tool items are absent altogether. A flat estimate computed on such a context has two possible sources that must be distinguished: the agent may not track the horizon, or the evidence it would need may simply be missing from what it was shown. The first is the phenomenon this paper measures; the second would be an artifact of our reconstruction.

We separate them with a fourth condition, the **tool arm**, which differs from local context in exactly one respect: the history is rebuilt without abbreviation, so every tool call and every tool result appears at full length and the harness that omitted them includes them. Nothing else changes — same sessions, same frozen steps, same questions, same parser. The arm is replay-faithful rather than interactive: the agent sees everything it saw, in full, but cannot choose to look somewhere new. Genuinely interactive probing would need the workspace as it stood at that turn, and with roughly a third of tool calls being shell commands that is not faithfully reconstructable.

Over 39 sessions probed in both conditions, across two tasks and all five agents:

	Calibration slope		O:E ratio		Skill	
Measure	Local	Tool	Local	Tool	Local	Tool
Remaining requirements	$+0.07$	$+0.14$	0.18	0.20	-0.61	-0.60
Remaining user turns	$+0.04$	$+0.06$	0.29	0.30	-0.41	-0.38
Remaining wall clock	$+0.27$	$+0.31$	0.88	0.84	$+0.01$	-0.01

The improvement is consistent in direction and negligible in size. Per session, the calibration slope rises by a median of $+0.036$ for remaining requirements and $+0.031$ for remaining user turns, improving in 69% and 72% of sessions respectively — a reliable effect, against the 1.0 that tracking would require. Under the classification of Section 3.1 the count measures do not move at all: remaining requirements is flat in 90% of sessions under both conditions, and remaining user turns is flat in 100% of sessions under both. Wall clock is the one measure whose shape responds, with five sessions crossing from flat prediction into loose tracking ($31\% \rightarrow 44\%$) and two leaving over-prediction. Its two skill figures, $+0.01$ and -0.01, straddle zero and should not be read as a direction: on this measure both conditions are indistinguishable from the leave-one-out median baseline.

That split is the informative part, and it is what we would expect if the myopia is about the user rather than about the repository. Tool output is evidence of what has already been *done*; it says nothing about what the user has yet to *ask for*. So the measure that depends on elapsed work — wall clock — improves slightly, and the measures that depend on the unrevealed remainder of the requirement list do not improve at all. Supplying the agent with its own full working record does not tell it what is coming, which is consistent with the global-context result in Section 3.2: what changes the count measures is being told the list, and nothing else we have tried changes them.

We report this arm on two of the five tasks rather than all five. It is an instrument check rather than a headline result, and the two tasks span both harness families — including the one whose reconstruction dropped tool items entirely, where any effect would have to be largest.

F.5 Ablation: Simulator Strength

Finding: replacing the simulated user with a frontier model from a different family changes the sessions — they run five minutes shorter — without changing calibration. The paired difference in slope is -0.015 locally, with a 95% interval of $[-0.033, +0.018]$, against a local-to-global gap of $+0.05$ to $+0.72$ on the same sessions.

The simulated user is a fixed model throughout the main results, which leaves open whether the agent's flat estimate is a property of the agent or of the user it was paired with. A weaker simulator could plausibly produce sessions that are harder to anticipate — less consistent in how it voices requirements, less predictable in when it accepts one — in which case the myopia would be a fact about our instrument rather than about coding agents.

We test this by varying the simulator while holding everything else fixed. Two tasks and two agents (`claude-opus-5` and `gpt-5-mini`) are each run under two simulators: the `qwen3.5-27b` used throughout, and `gemini-2.5-pro`. The frontier simulator is chosen from a third model family, rather than matching each agent to its own model, so that the comparison is about simulator capability and not about an agent recognising its own outputs; the matched-model variant is reported at the end of this section. Four repeats per cell give 32 sessions, all of which graded and were accepted; 31 are probed under both the local and global conditions.

The manipulation changed the sessions. Sessions under the frontier simulator are materially shorter in wall clock — median 24.0 minutes against 29.3, Mann–Whitney $p = 0.018$ — while reward (0.58 against 0.65, $p = 0.252$) and turn count (12.0 against 12.5, $p = 0.953$) are indistinguishable. The simulator therefore does affect the sessions being measured, which is what makes the calibration comparison informative rather than vacuous.

Calibration did not move. Paired over the 16 (task, agent, measure) strata:

	Calibration slope		O:E ratio	
Condition	Qwen3.5-27B	Gemini 2.5 Pro	Qwen3.5-27B	Gemini 2.5 Pro
Local	$+0.05$	$+0.01$	0.42	0.39
Global	$+0.72$	$+0.74$	0.97	0.96

Wilcoxon signed-rank tests give $p = 0.562$ on slope locally and $p = 0.528$ globally. On 16 strata a failure to reject is weak evidence by itself, so we report the interval rather than the test. The bootstrapped paired difference in slope (frontier minus Qwen) is -0.015 with a 95% interval of $[-0.033, +0.018]$ locally and $+0.005$ with $[-0.034, +0.090]$ globally. The local interval excludes any simulator effect larger than about 0.035 in either direction; the global interval is wider but still admits at most $+0.09$. Both are an order of magnitude below the change the requirement list produces on these same sessions, $+0.05$ to $+0.72$, and more than an order below the 1.0 a calibrated estimate would reach. Under the classification of Section 3.1, wall-clock estimates in the local condition are Flat Predictions in 15 of 15 sessions under the Qwen simulator and 13 of 15 under the frontier simulator.

A matched simulator. A smaller run drives the simulator with the same model as the agent under test, rather than a third family. It answers a weaker question — whether the two sides of a session differ in estimation error, not whether simulator strength changes the agent — and the answer is that they do not: mean absolute error is 6.35 for the agent against 6.74 for the simulator on remaining user turns, and 6.54 against 6.53 on remaining agent turns, over $1{,}433$ scored pairs. We report it only because matching the models is the first thing a reader is likely to suggest; the comparison above is the one that isolates simulator strength, because matching confounds capability with the agent recognising its own outputs.

Scope. This is run on two of the five tasks and two of the five agents, and `remaining_tokens` is omitted because recovering its truth requires a rescoring pass we did not run on this set. It is therefore a bounded check on one confound rather than a replication of the main grid. What it establishes is that the effect of simulator strength on calibration is small enough to be excluded as an explanation of the myopia, not that it is exactly zero.