FRAGILE FOLLOW-THROUGH: UNCOVERING HOW USER REVISIONS DISRUPT LONG-HORIZON TASK EXECUTION IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents are increasingly entrusted with complex, long-horizon tasks, where fulfilling a user's request requires coordinating decisions and actions across many interdependent steps. Existing work typically evaluates agents against requirements specified at the outset, although recent studies have begun to consider user interruptions and revisions during execution. However, current evaluations do not adequately capture revisions prompted by task progress or explain how they disrupt long-horizon task completion. To bridge this gap, we adapt long-horizon tasks by altering their initial instructions and introducing mid-execution revisions that restore the original requirements. This yields REQUEST, an application-grounded benchmark designed to evaluate task completion and analyze agent behavior under dynamic user revisions. Revisions are triggered following observable progress under the altered requirements, with the agent's history and environment state preserved. Extensive experiments show that user revisions reduce average task success rates by approximately 17–63% relative to the original tasks across models. Process analyses help explain these losses: agents retain incompatible prior decisions, lose valid progress, or leave unchanged requirements unmet. Answers and guidance targeting these problems yield uneven benefits, pointing to mixed difficulties behind fragile follow-through in long-horizon tasks. Our benchmark, code, and evaluation protocols will be publicly available to support reproducible evaluation.

1 INTRODUCTION

The growing capabilities of LLMs have spurred efforts to build agents capable of executing complex tasks on users' behalf (Zhou et al., 2024; Xu et al., 2025). Real-world requests often combine multiple requirements that depend on a sequence of interconnected decisions (Yao et al., 2025; Zhang et al., 2026a). An early choice can dictate subsequent information gathering and constrain future actions. Reliable delegation, therefore, demands that these decisions remain strictly aligned with user requirements throughout execution until the intended outcome is achieved.

Current task-oriented evaluations primarily assess whether agents achieve requested outcomes (Liu et al., 2024; Mialon et al., 2024; Wei et al., 2025). For long-horizon tasks, this typically involves verifying environment states or joint constraint satisfaction (Zhou et al., 2024; Zhang et al., 2026a). Intermediate progress metrics further track completed subgoals and state changes, highlighting partial fulfillment even when the overall task remains unfinished (Ma et al., 2024; Lu et al., 2025; Xu et al., 2025). While most evaluations assume static requirements throughout execution, recent studies have begun to explore mid-task corrections and dynamically revealed constraints (Liu et al., 2026; Zou et al., 2026).

However, current evaluations do not adequately capture revisions prompted by task progress or explain how these changes disrupt long-horizon task completion. In practice, users often recognize mistaken assumptions or reconsider their requirements as execution unfolds. Such revisions can force agents to revisit earlier decisions, adapt subsequent choices, and manage ripple effects that extend beyond the explicitly modified requirement. As illustrated in Figure 1, an agent might successfully execute a revised step but simultaneously fail to preserve prior, still-valid work. Understanding

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.

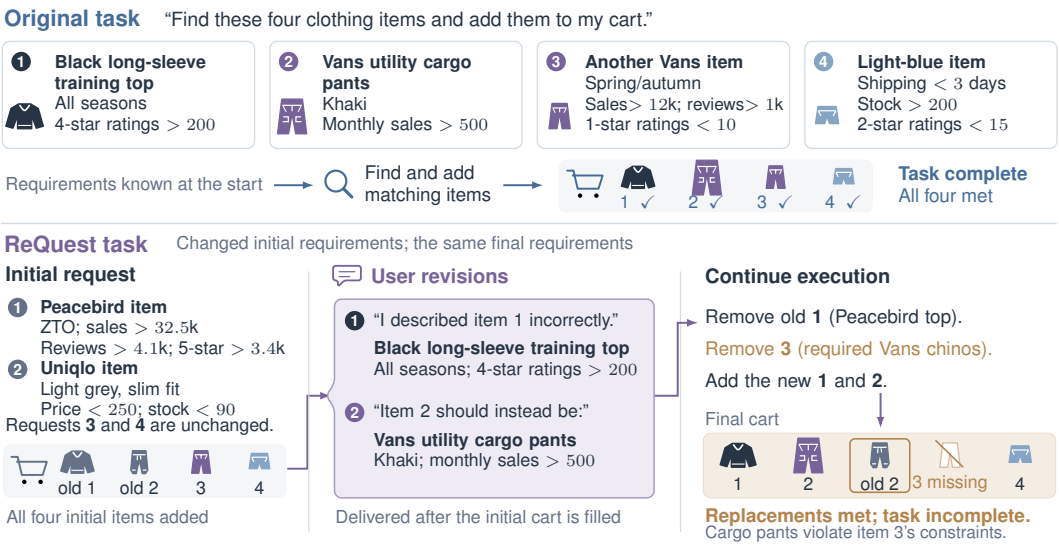


Figure 1: **User revisions can disrupt long-horizon task completion.** Both executions have the same final requirements. The original task succeeds, whereas the ReQuest task adds both replacements but removes a still-required item. Please refer to Appendix H.1 for more details.

these failures requires distinguishing difficulty in solving revised subproblems from difficulty in reconciling their results with prior decisions and completing the remaining work.

To address this gap, we introduce REQUEST, a benchmark spanning deep search, web navigation, and structured planning. Tasks begin with altered instructions and receive revisions that restore the original requirements after observable progress, enabling evaluation against the same outcome criteria as the original tasks. Evaluation of these task pairs first quantifies changes in task completion. Independent subproblem evaluations, error attribution, and requirement-level analyses then diagnose where execution breaks down. Finally, targeted support interventions test whether answers or guidance can address these difficulties. Together, these analyses aim to explain how revisions interact with task progress and interdependent decisions to affect long-horizon task completion.

Extensive experiments reveal that user revisions reduce average task success rates by approximately 17–63% compared to the original tasks across LLM backbones. Beyond degrading individual model performance, revisions can transform previously solved tasks into failures shared across all tested models. Independent evaluations show that separate solving does not ensure completion under revisions. Error attribution and requirement-level analyses help explain this gap by revealing incompatible choices, lost valid progress, and unmet requirements beyond the corrected subproblem. Support interventions targeting these difficulties yield mixed results, with answers and guidance helping in some settings but not consistently restoring completion. Together, these findings expose mixed difficulties behind fragile follow-through in long-horizon tasks.

Our key contributions are:

- **A dynamic benchmark for task completion under user revisions.** REQUEST adapts long-horizon tasks across three domains, delivering mid-execution corrections after observable progress. It enables paired comparisons with original tasks under the same final requirements and outcome evaluators.
- **A rigorous characterization of agent sensitivity to revisions.** Evaluations across ten diverse open-weight and proprietary LLM backbones quantify declines in task completion and coverage, revealing both model-specific losses and shared failures.
- **Insights for improving decision updates and task follow-through.** Independent subproblem evaluations, error attribution, and requirement-level analyses expose difficulties in updating decisions and preserving valid progress. The uneven benefits of targeted support motivate future work on integrating revised solutions with still-valid work and carrying long-horizon tasks through to completion.

2 Related Work

Long-horizon LLM agent evaluation. Long-horizon agent benchmarks assess agents' ability to complete complex tasks (Zhou et al., 2024; Xu et al., 2025). Success in these settings depends on the resulting environment state and the joint satisfaction of task requirements, extending evaluation beyond the correctness of individual responses (Zhang et al., 2026a). Measures of intermediate progress and execution behavior complement these completion-based assessments by identifying where and how agents fall short (Ma et al., 2024; Lu et al., 2025; Cai et al., 2026). These outcome and process assessments generally use the requirements specified at the outset as their reference.

Multi-turn interaction and task adaptation. Research on multi-turn interaction evaluates instruction following across dialogue turns (He et al., 2024) and task completion through ongoing interaction with users (Yao et al., 2025; Lu et al., 2024). Even when the intended task remains fixed, distributing its requirements across turns can reduce reliability (Laban et al., 2026). When the problem itself changes, work on interruptible reasoning examines response accuracy under mid-inference updates to mathematics and programming tasks (Wu et al., 2026). In interactive planning, AdaPlanBench evaluates replanning in response to feedback that reveals previously hidden world and user constraints (Liu et al., 2026). Moving from newly revealed constraints to revised goals, AgentChangeBench assesses adaptation to mid-dialogue goal shifts in enterprise workflows through task success, tool-use efficiency, and recovery time (Rana et al., 2025). In long-horizon web environments, InterruptBench evaluates the benefits of user updates delivered at predefined fractions of a baseline trajectory (Zou et al., 2026). Taken together, these studies emphasize the incorporation of new information and recovery from changed instructions. Less attention has been paid to how revisions prompted by task progress affect interdependent decisions and subsequent task completion.

Long-horizon agent reliability. Reliable execution requires agents to maintain relevant goals and information as their work develops. Studies of goal drift examine departures from assigned objectives over extended interactions (Arike et al., 2025), while work on evolving information environments tests whether agents update beliefs when new evidence overturns earlier conclusions (Ji et al., 2026). Both lines of work connect reliability to the retention and updating of task information. To support these processes, proposed dialogue runtimes make goal revision and resumption explicit (Garralda-Barrio, 2026), and state-management approaches retain verified progress across execution contexts (Ma et al., 2026). These approaches highlight the contribution of the execution scaffold to reliable task completion (Zhang et al., 2026b).

3 ReQuest

ReQuest is designed to examine how user revisions interact with task progress and dependencies to affect long-horizon task completion. Each ReQuest task is constructed by altering the initial requirements of an existing task and restoring them through progress-triggered corrections. The formulation below distinguishes dependencies among task decisions from prerequisites for delivering revisions, providing the basis for benchmark construction and comparison with the original tasks.

3.1 Task Formulation

The interaction begins with a user request x_0, whose requirements $\mathbf{c}_0$ guide the agent's initial work. During execution, the user corrects or reconsiders individual requirements through further messages, termed *revisions*. The agent retains its conversation history and environment state, including the effects of earlier decisions. A *ReQuest task* comprises this complete interaction: the initial request, intermediate corrections, and subsequent actions under revised requirements.

Dependencies among constituent task decisions can be represented with a directed acyclic graph $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$. At node v_i, the agent produces a result or choice b_i under requirement c_i. Outgoing edges connect it to decisions that depend on this result. Let $\mathcal{P}_i = \{k : (v_k, v_i) \in \mathcal{E}_T\}$ index its immediate predecessors. In this abstraction, the Boolean predicate ϕ_i represents satisfaction of the local requirement and compatibility with predecessor results $\mathbf{b}_{\mathcal{P}_i}$. A consistent solution satisfies

$$\phi_i(b_i, \mathbf{b}_{\mathcal{P}_i}; \mathbf{c}) = 1, \qquad \forall v_i \in \mathcal{V}_T. \tag{1}$$

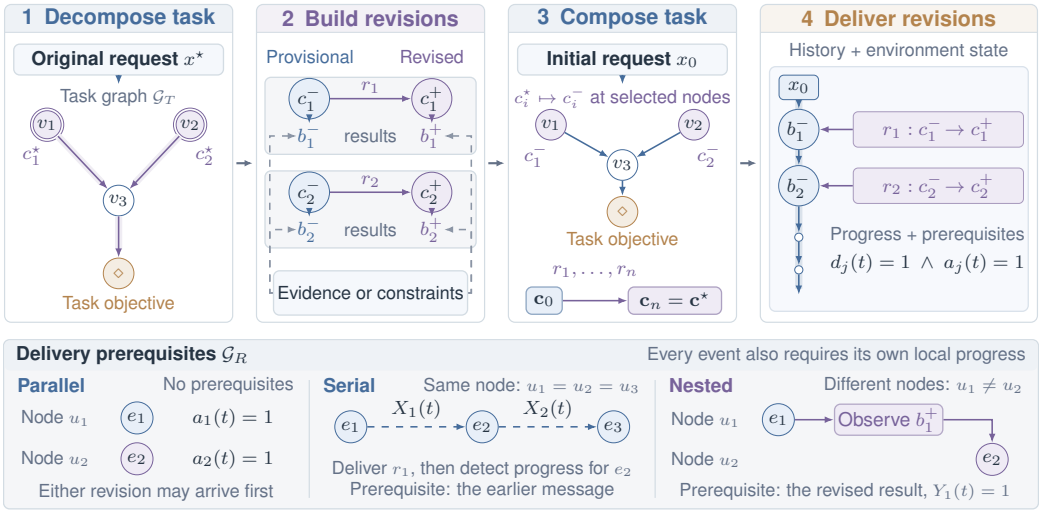


Figure 2: **Constructing progress-triggered user revisions.** Selected requirements of an original task are altered in the initial request and restored through subsequent corrections. Task progress and revision dependencies determine when each correction arrives.

The vectors $\mathbf{b}$ and $\mathbf{c}$ collect the node values and current requirements. A revision event e_j targets an existing node $u_j \in \mathcal{V}_T$, and several events may share the same target. Its message r_j replaces a provisional requirement c_j^- with a revised requirement c_j^+. These superscripted values are indexed by revision event, whereas c_i denotes the current requirement at node v_i. The local problems of finding results that satisfy c_j^- and c_j^+ are termed the provisional and revised *subproblems*, respectively. The agent must resolve the revised requirement and reconcile its result with dependent work. Still-valid choices can be retained, while shared constraints may require adjustments elsewhere in the task.

Revision events form a second directed acyclic graph, $\mathcal{G}_R = (\mathcal{V}_R, \mathcal{E}_R)$, whose nodes are revision events and whose edges specify prerequisites for delivering them. The mapping $e_j \mapsto u_j$ connects the two graphs. Let $\mathcal{Q}_j = \{k : (e_k, e_j) \in \mathcal{E}_R\}$ index the immediate predecessors of e_j. The binary indicators $X_j(t)$ and $Y_j(t)$ record whether message r_j has been delivered and its revised result b_j^+ has subsequently been detected by step t, respectively. For an undelivered event, its prerequisites are satisfied when $a_j(t) = 1$, where

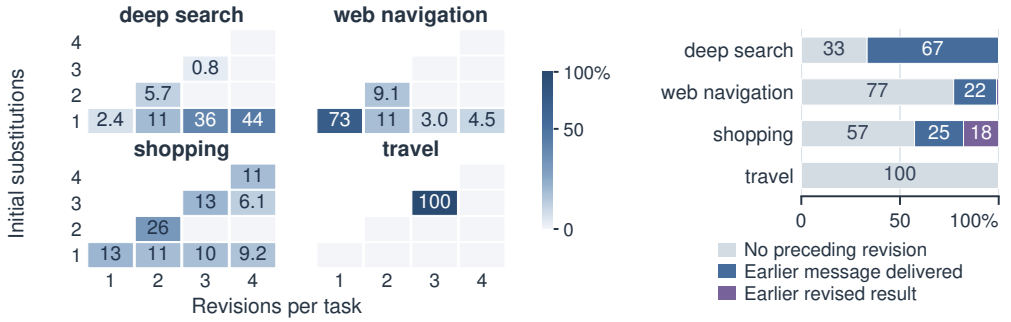
$$a_j(t) = \prod_{\substack{k \in \mathcal{Q}_j \\ u_k = u_j}} X_k(t) \prod_{\substack{k \in \mathcal{Q}_j \\ u_k \neq u_j}} Y_k(t). \tag{2}$$

Serial links revise the same task node ($u_k = u_j$) and require $X_k(t) = 1$: the earlier message introduces the condition under which progress can prompt another correction. *Nested* links connect different nodes ($u_k \neq u_j$) and require $Y_k(t) = 1$, so a later correction waits for detection of the earlier revised result. *Parallel* revisions have no delivery dependencies on one another. For any event without prerequisites, $\mathcal{Q}_j = \varnothing$ and $a_j(t) = 1$, since empty products equal one. Thus, $\mathcal{G}_T$ describes dependencies in the work itself, while $\mathcal{G}_R$ governs when corrections can follow one another. Each event also requires progress under the condition it changes, as described below.

3.2 Benchmark Construction

The source benchmarks capture realistic long-horizon applications involving interdependent decisions and actions. As illustrated in Figure 2, an *original task*, with instruction $x^\star$, requirements $\mathbf{c}^\star$, and evaluator V, is transformed into an initial request and corrections linked to the agent's progress.

Task decomposition and revision construction. With LLM assistance, intermediate decisions and their dependencies in the original task are first identified. Nonterminal nodes whose changed results affect subsequent work are then selected for revision. For each revision event e_j at a selected



(a) Initial substitutions and subsequent revisions.

(b) Revision prerequisites.

Figure 3: **Task variation and revision dependencies.** (a) Joint distribution of initial text substitutions and subsequent revisions, as percentages of tasks within each subset. (b) Shares of events requiring no prior revision, an earlier message, or an earlier revised result.

node u_j, the provisional and revised requirements c_j^- and c_j^+ are grounded in evidence or executable constraints. Their corresponding results are denoted b_j^- and b_j^+.

These local substitutions are composed jointly into one initial request x_0 and a set of user messages $\{r_j : e_j \in \mathcal{V}_R\}$. The substituted requirements and their dependencies in $\mathcal{G}_R$ jointly determine which conditions appear initially and which are introduced by subsequent corrections. A message can restore an original condition or introduce the next provisional problem in a serial chain. Messages state the local change without supplying its solution, restating the whole task, or instructing the agent to resume.

The corrections must jointly recover the original requirements so that both task versions can use the same outcome evaluator. With $n = |\mathcal{V}_R|$ revisions and $\mathbf{c}_k$ denoting the requirements after the first k messages, this requires $\mathbf{c}_n = \mathbf{c}^\star$ for every valid dependency order.

Revision delivery during execution. Each revision is triggered by observable progress under the requirement being changed. For event e_j, a detector identifies the provisional result b_j^- in the agent's messages or action arguments, indicating that the agent has expressed or used this result after the provisional requirement takes effect. Let $d_j(t) \in \{0, 1\}$ indicate whether such progress has been detected by step t. The message is delivered once both progress and prerequisites are established:

$$\tau_j = \min\{t : a_j(t) = d_j(t) = 1\}. \tag{3}$$

If the conditions are never met, $\tau_j = \infty$. Ambiguous references are resolved by an LLM judge using the surrounding context. Retrieved text and tool responses alone do not trigger delivery. The correction enters as the next user message, preserving the conversation and any environment changes made by the triggering action. These observations govern revision delivery, while the outcome evaluator assesses task completion.

Application coverage and construction statistics. The resulting application subsets are deep search, adapted from BrowseComp (Wei et al., 2025); web navigation, from WebArena (Zhou et al., 2024); and structured planning, from DeepPlanning (Zhang et al., 2026a). The structured planning subset comprises shopping and travel task families. These settings cover corrections to evidence chains, the entities targeted by website actions, and coupled planning choices, respectively.

The resulting benchmark contains 359 original–ReQuest task pairs: 123 in deep search, 66 in web navigation, and 170 in structured planning (98 shopping and 72 travel). Together, they contain 962 revision events. Figure 3 relates the number of initial text substitutions to subsequent revisions and summarizes their delivery prerequisites. These constructions cover repeated changes to one requirement and corrections across several dependent decisions. Serial revisions predominate in deep search, repeatedly redirecting an evidence chain, while web navigation mainly uses single corrections to action targets. Shopping includes nested revisions in which a revised choice shapes later decisions. Travel revisions can be delivered independently while jointly constraining the same plan. Independent delivery therefore need not imply independent planning decisions.

Task validation. Automated validation checks subproblem solutions against evidence, executable constraints, or website outcomes. Models obtain these solutions with task-specific hints unavailable during evaluation. Required irreversible operations are checked to ensure that the final task remains feasible and that completion can still be assessed using the original evaluator.

Composition checks verify that corrections preserve unaffected requirements and restore the original specification, with valid, acyclic revision dependencies. Two human annotators then review all automatically constructed tasks passing these checks for scenario plausibility and clear, coherent corrections. They manually refine or correct 31% of these tasks to produce the final benchmark. Please refer to Appendix C for validation and delivery checks.

3.3 Evaluation Protocol

The evaluation protocol combines final task outcomes with revision coverage. Each original–ReQuest pair is assessed using the same outcome evaluator V, while delivery records establish whether the ReQuest execution encounters the full sequence of corrections.

Whole-task success and revision coverage. For each of N task pairs, indexed by s, let $S_o(s)$ and $S_r(s)$ be binary outcome indicators for the original and ReQuest executions: each equals one if V judges the final outcome successful, and zero otherwise. To count a ReQuest execution as successful, the agent must also have encountered the complete sequence of planned corrections. Let $F_s = 1$ when every revision has been delivered and any revised results required for subsequent delivery have been observed. Otherwise, $F_s = 0$. The original-task success rate SR_o is the fraction of original tasks successfully completed. The ReQuest-task success rate SR_r additionally requires full revision coverage and uses the same population of N tasks:

$$\mathrm{SR}_o = \frac{1}{N}\sum_{s=1}^{N} S_o(s), \qquad \mathrm{SR}_r = \frac{1}{N}\sum_{s=1}^{N} F_s S_r(s). \tag{4}$$

The relative success-rate drop is $(\mathrm{SR}_o - \mathrm{SR}_r)/\mathrm{SR}_o$ as a percentage when $\mathrm{SR}_o > 0$. For the cross-domain average, each success rate is first averaged equally across the three domains, and the relative drop is then computed from these two averages.

To distinguish incomplete revision coverage from failure after full coverage, the full revision delivery rate is $\mathrm{DR} = n_F/N$, where $n_F = \sum_{s=1}^{N} F_s$. Post-Revision Completion (PRC) measures success among these fully covered executions:

$$\mathrm{PRC} = \frac{\sum_{s=1}^{N} F_s S_r(s)}{n_F}, \qquad \mathrm{SR}_r = \mathrm{DR} \cdot \mathrm{PRC}. \tag{5}$$

This decomposition shows whether low SR_r accompanies limited revision coverage, low completion after coverage, or both. DR indicates the fraction of tasks included in PRC. PRC is undefined when $n_F = 0$. Because coverage depends on execution, its task population can differ across agents.

Please refer to Appendices D and E for evaluator details and statistical procedures.

4 Experiments

4.1 Experimental Setup

With the benchmark introduced above, LLM agents are evaluated on original and ReQuest tasks to measure changes in task completion and examine how execution unfolds under user revisions.

Agent scaffold and LLM backbones. Evaluation uses smolagents (Roucher et al., 2025) for the deep search subset, AgentLab (Drouin et al., 2024; de Chezelles et al., 2025) for web navigation, and Qwen-Agent (Qwen Team, 2026d) for structured planning. Ten LLM backbones are evaluated: Qwen3.6-35B-A3B, Qwen3.8-Flash, Qwen3.8-Max (Qwen Team, 2026a;b;c), DeepSeek-V4-Flash (DeepSeek-AI, 2026), Gemini-3.7-Flash (Google DeepMind, 2026), GLM-5.3-Flash (GLM-5-Team et al., 2026), GPT-OSS-20B and GPT-OSS-120B (OpenAI, 2025), and GPT-5.6-Luna and GPT-5.6-Sol (OpenAI, 2026).

Table 1: **Agent performance on original tasks and ReQuest across three domains.** All values are percentages. Darker shading indicates greater negative effects of revisions.

Model	Deep Search					Web Navigation					Structured Planning					Average		
	Orig.	ReQuest				Orig.	ReQuest				Orig.	ReQuest				Orig.	ReQuest	
	$SR_o\uparrow$	DR↑	$SR_r\uparrow$	PRC↑	Drop↓	$SR_o\uparrow$	DR↑	$SR_r\uparrow$	PRC↑	Drop↓	$SR_o\uparrow$	DR↑	$SR_r\uparrow$	PRC↑	Drop↓	$SR_o\uparrow$	$SR_r\uparrow$	Drop↓
Qwen3.6-35B-A3B	36.6	56.1	9.8	17.4	73.3	48.5	43.9	22.7	51.7	53.1	30.0	80.0	14.1	17.6	52.9	38.4	15.5	59.5
Qwen3.8-Flash	74.8	87.0	48.0	55.1	35.9	75.4	98.5	69.2	70.3	8.2	58.8	87.6	48.2	55.0	18.0	69.7	55.1	20.8
Qwen3.8-Max	78.0	87.8	58.5	66.7	25.0	75.8	97.0	69.7	71.9	8.0	70.0	91.8	58.2	63.5	16.8	74.6	62.2	16.7
DeepSeek-V4-Flash	82.1	78.9	48.0	60.8	41.6	72.7	95.5	60.6	63.5	16.7	58.8	87.6	47.6	54.4	19.0	71.2	52.1	26.9
Gemini-3.7-Flash	65.0	76.4	26.0	34.0	60.0	72.7	62.1	45.5	73.2	37.5	34.7	71.2	27.1	38.0	22.0	57.5	32.8	42.9
GLM-5.3-Flash	59.3	59.3	34.1	57.5	42.5	78.5	100.0	75.4	75.4	3.9	56.5	88.8	45.9	51.7	18.8	64.8	51.8	20.0
GPT-OSS-20B	27.6	52.0	0.8	1.6	97.1	43.9	56.1	22.7	40.5	48.3	17.6	69.4	9.4	13.6	46.7	29.7	11.0	63.1
GPT-OSS-120B	30.9	52.8	8.1	15.4	73.7	62.1	78.8	43.9	55.8	29.3	22.9	82.9	12.9	15.6	43.6	38.7	21.7	43.9
GPT-5.6-Luna	52.0	65.9	22.8	34.6	56.2	56.9	56.9	33.8	59.5	40.5	29.4	71.2	13.5	19.0	54.0	46.1	23.4	49.3
GPT-5.6-Sol	81.3	82.9	63.4	76.5	22.0	86.4	97.0	72.7	75.0	15.8	67.1	85.9	50.6	58.9	24.6	78.2	62.2	20.4



(a) deep search. (b) web navigation. (c) structured planning.

Figure 4: **User revisions reduce cross-model task coverage.** Share of tasks solved by at least k of ten models. Web navigation uses 64 shared tasks.

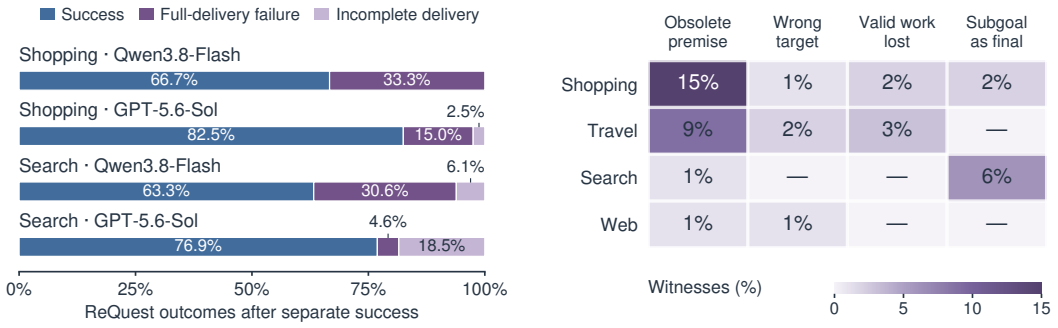
Evaluation settings. Sampling and reasoning settings are chosen per backbone. Each backbone's original and ReQuest runs use the same scaffold and retain these settings. Deep search allows up to 200 turns for original tasks and 250 for ReQuest. Structured planning allows up to 400 turns per shopping phase and 150 per travel task. Web navigation uses identical task-specific limits of up to 200 turns for original and ReQuest runs. Web results use 65 task pairs for Qwen3.8-Flash, GLM-5.3-Flash, and GPT-5.6-Luna, and 66 for other backbones. Structured-planning success rates pool shopping and travel tasks using their respective evaluators.

4.2 Task Completion under User Revisions

User revisions substantially reduce agents' task completion rates. As shown in Table 1, relative losses in average success range from 17–63% across backbones. GPT-5.6-Sol and Qwen3.8-Max, the strongest original-task backbones, show relative losses of 20.4% and 16.7%, respectively. For GPT-OSS-20B and Qwen3.6-35B-A3B, the losses reach 63.1% and 59.5%. The losses span capability levels: strong original-task performance does not ensure robustness to revisions, while weaker backbones can suffer larger declines. Mean endpoint losses across models are 18.5 percentage points (pp) in deep search, 14.4 pp in web navigation, and 11.6 pp in structured planning.

Beyond each model's average loss, cross-model task coverage measures how many backbones solve each task. The share of tasks solved by at least k models reveals whether revisions narrow this coverage. As illustrated in Figure 4, the share solved by at least six models falls from 58% to 16% in deep search. Across domains, the share solved by none rises from 9% to 17%. A comparison of individual tasks identifies 36 tasks solved by at least one model originally but by none under ReQuest. Revisions can therefore turn tasks with observed original success into failures shared by every tested backbone.

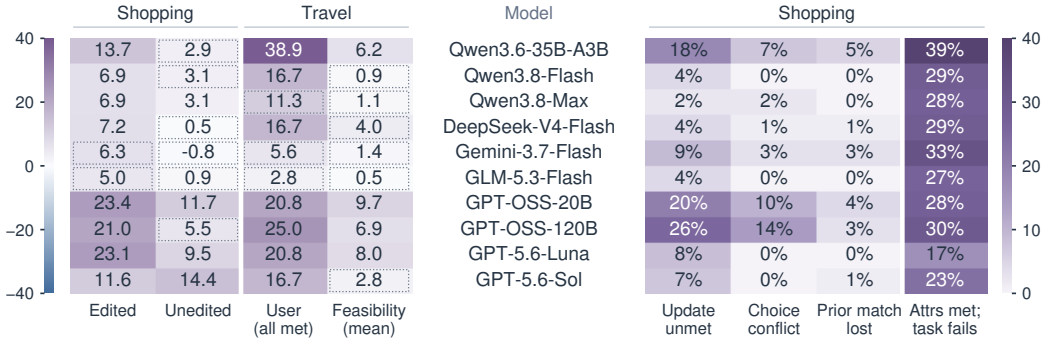
Together, individual losses and failures shared across backbones show that original-task performance alone is insufficient to characterize reliability under changing requirements.



(a) Outcomes after separate success.

(b) Integration error witnesses.

Figure 5: **Separate solving and errors during revision.** (a) ReQuest outcomes when the original task and every independent subproblem succeed. (b) Confirmed witnesses per 100 sampled executions in each task family. Categories overlap. Dashes mark no retrieved witness.



(a) Reference matches and planning scores.

(b) State errors and incomplete outcomes.

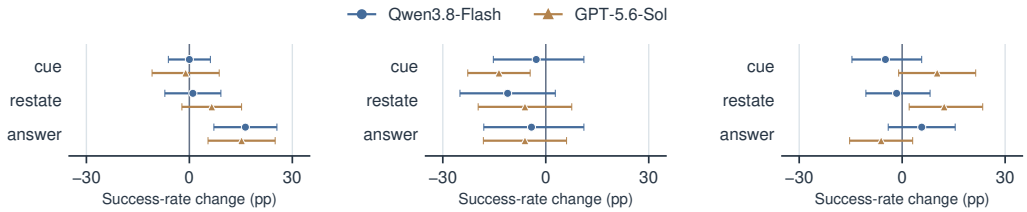
Figure 6: **Reference-product matches, planning scores, and state errors.** (a) Original − ReQuest (pp); dotted borders: 95% CI covers 0. (b) Confirmed shares (%), with overlap; GPT-5.6-Luna: 24 tasks; others: 98.

4.3 Diagnosing Failures under User Revisions

The completion losses above motivate examining how agents reconcile revised requirements with prior decisions and task progress to complete the whole task.

Separate solving is insufficient. Each provisional and revised subproblem is first executed independently with the corresponding tools, using Qwen3.8-Flash and GPT-5.6-Sol in shopping and deep search. As illustrated in Figure 5a, even when the original task and all subproblems succeed separately, 5–33% fail under ReQuest after full revision delivery.

Updating decisions and preserving progress. To explain why separate solving does not ensure successful revised execution, 400 ReQuest trajectories are sampled equally across four task families. LLM-assisted error attribution followed by human review identifies errors in handling prior decisions and task progress. As illustrated in Figure 5b, agents may retain an obsolete price ceiling that excludes a feasible product in shopping, apply a meal correction to another required meal and lose valid work in travel planning, or terminate with an intermediate answer in deep search. These errors concern how a correction is incorporated into ongoing work. A superseded condition can continue to constrain later choices, while an update applied to the wrong target can leave the intended change unmet and remove valid work elsewhere. Terminating with an intermediate answer creates a further gap between resolving the revised subproblem and satisfying the user's overall request. Their consequences can therefore extend beyond the requirement explicitly changed.



(a) shopping. (b) travel. (c) deep search.

Figure 7: **Task completion under revision support.** Points show changes from matched ReQuest controls (pp), with 95% task-bootstrap intervals. Qwen3.8-Flash and GPT-5.6-Sol are evaluated in all three task families.

Effects beyond the corrected subproblem. To assess whether revision-related losses extend beyond the corrected subproblem, reference-product match rates for edited and unedited shopping slots are compared alongside joint user-constraint satisfaction and average feasibility in travel. As illustrated in Figure 6a, the reference-product match rate for unedited slots falls by up to 14.4 pp. Feasibility also declines, although several intervals include zero. Reconstructing carts from successful tool operations further connects local progress to whole-task outcomes. Figure 6b shows unmet updates, incompatible selections, and lost progress. Even after full delivery, 17–39% of evaluated tasks per model satisfy the checked revised attributes but fail overall. These checks exclude coupons and prices, which remain separate obligations.

These analyses reveal fragile follow-through: agents can solve constituent problems separately yet falter when requirements change. Their errors involve inconsistent updates to earlier decisions, lost valid progress, and unmet requirements beyond the corrected subproblem.

4.4 Effects of Revision Support

To probe these difficulties, *answer* supplies available subproblem answers to ease local solving, *restate* makes current requirements explicit to aid their reconstruction, and *cue* prompts continuation of the overall task. Each comparison matches support and standard ReQuest runs by task and backbone, retaining the same inference settings, tools, and budgets.

These supports yield uneven benefits. As illustrated in Figure 7, answer raises shopping completion by 15.2–16.3 pp across both backbones, showing that access to subproblem solutions can help. Its benefit remains unclear in travel and deep search. Restate helps in the deep-search replication, but yields no clear shopping or travel benefit. Cue shows no clear gain and lowers completion in the travel replication. Providing a local answer still leaves the agent to incorporate it into prior decisions. Likewise, restating requirements or requesting continuation does not itself repair incompatible choices or recover valid work already lost. This helps explain why support can improve completion without consistently restoring it.

Together, support comparisons and error attribution point to mixed difficulties behind fragile follow-through. Targeted answers and restated requirements can help, yet agents also struggle to update related decisions, preserve still-valid progress, and complete the whole task.

5 Conclusion

We introduce ReQuest, a benchmark for examining how user revisions disrupt long-horizon task execution. Across backbones, revisions reduce task completion, but local solving difficulty does not fully explain these losses: failures persist even when the original task and all subproblems succeed separately. Trajectory analysis helps explain this gap: in handling prior decisions and task progress, agents retain incompatible choices, lose valid progress, or leave requirements beyond the corrected subproblem unmet. Attempts to address these difficulties through answers or guidance improve completion in some settings, but leave substantial failures unresolved. Overall, fragile follow-through in LLM agents reflects mixed difficulties in solving revised problems, updating prior decisions, and preserving task progress to complete long-horizon tasks.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic datasets, implementing methods, cleaning and reformatting datasets, and supporting qualitative error attribution. We have not used generative AI tools for developing conceptual frameworks, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, or interpreting results, and developing theoretical models, formulating mathematical claims, providing critical ingredients for proofs, and assisting in proof writing are not applicable to this work. Additionally, we used generative AI tools for writing and editing code and polishing the manuscript text and the visual presentation of figures. We have reviewed all AI-assisted work. Two authors reviewed all automatically validated tasks and all trajectories selected for error attribution. The initial manuscript and figure designs were produced by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

REQUEST supports the study of agent reliability under changing user requirements. Its evaluations use public web information, self-hosted benchmark websites, and benchmark-provided product and travel data. Two authors conduct the manual review of task quality and agent trajectories. This annotation work did not undergo institutional ethics review. The benchmark may inherit biases from its source datasets. Appendix I lists the resource licenses and usage terms. Public artifacts will follow the corresponding redistribution terms, with access credentials and annotator-identifying information excluded.

REPRODUCIBILITY STATEMENT

To support reproducibility, Sections 3.2 and 3.3 specify benchmark construction and evaluation, while Appendix C documents validation and human review. The agent scaffolds, model configurations, execution settings, and scoring procedures are described in Section 4.1 and Appendix D. Appendices F and G detail the diagnostic procedures and revision-support comparisons. The supplementary materials include revision records, source-task identifiers, and code for revision delivery and completion metrics. Original benchmark assets are obtained from their respective releases. The benchmark, code, and evaluation protocols will be publicly released.

REFERENCES

Rauno Arike, Elizabeth Donoway, Henning Bartsch, and Marius Hobbhahn. Evaluating Goal Drift in Language Model Agents. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 8(1):192–203, October 2025. ISSN 3065-8365. doi: 10.1609/aies.v8i1.36541. URL http://dx.doi.org/10.1609/aies.v8i1.36541.

Yuandao Cai, Yuzhang Zhu, Liyou Gao, Wensheng Tang, and Shengchao Qin. Push Your Agent: Measuring and Enforcing Quantitative Goal Persistence in Long-Horizon LLM Agents, 2026. URL https://arxiv.org/abs/2605.23574.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The BrowserGym Ecosystem for Web Agent Research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3. Expert Certification.

DeepSeek-AI. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How Capable are Web Agents at Solving Common Knowledge Work Tasks? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp

(eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, January 1979. ISSN 0090-5364. doi: 10.1214/aos/1176344552. URL http://dx.doi.org/10.1214/aos/1176344552.

Mariano Garralda-Barrio. From Task-Guided Conversational Graphs to Goal-Oriented Dialogue Runtimes, 2026. URL https://arxiv.org/abs/2606.23797.

GLM-5-Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. GLM-5: from Vibe Coding to Agentic Engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Google DeepMind. Gemini 3.7 Flash: Model card, August 2026. URL https://deepmind.google/models/model-cards/gemini-3-7-flash/.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, Shruti Bhosale, Chenguang Zhu, Karthik Abinav Sankararaman, Eryk Helenowski, Melanie Kambadur, Aditya Tayade, Hao Ma, Han Fang, and Sinong Wang. Multi-IF: Benchmarking LLMs on Multi-Turn and Multilingual Instructions Following, 2024. URL https://arxiv.org/abs/2410.15553.

Haonian Ji, Kaiwen Xiong, Siwei Han, Peng Xia, Shi Qiu, Yiyang Zhou, Jiaqi Liu, Jinlong Li, Bingzhou Li, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. ClawArena: Benchmarking AI Agents in Evolving Information Environments, 2026. URL https://arxiv.org/abs/2604.04202.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs Get Lost In Multi-Turn Conversation. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 54738–54778, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/59f6421e64707225fdf5b28840679a07-Paper-Conference.pdf.

Jiayu Liu, Cheng Qian, Zhenhailong Wang, Bingxuan Li, Jiateng Liu, Qing Zong, Heng Wang, Jeonghwan Kim, Yumeng Wang, Bingxiang He, Xiusi Chen, Yi R. Fung, and Heng Ji. AdaPlan-

Bench: Evaluating Adaptive Planning in Large Language Model Agents under World and User Constraints, 2026. URL https://arxiv.org/abs/2606.05622.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/e9df36b21ff4ee211a8b71ee8b7e9f57-Paper-Conference.pdf.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A Stateful, Conversational, Interactive Evaluation Benchmark for LLM Tool Use Capabilities. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Xing Han Lu, Zdeněk Kasner, and Siva Reddy. WebLINX: Real-World Website Navigation with Multi-Turn Dialogue. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 33007–33056. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/lu24e.html.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 74325–74362. Curran Associates, Inc., 2024. doi: 10.52202/079017-2365. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/877b40688e330a0e2a3fc24084208dfa-Paper-Datasets_and_Benchmarks_Track.pdf.

Ziyu Ma, Hailang Huang, Shun Zou, Yong Wang, Shidong Yang, Yiming Hu, Fei Wei, and XiangXiang Chu. LongHorizon-Harness: Advancing Long-Horizon Agents for Real-World Tasks, 2026. URL https://arxiv.org/abs/2608.01964.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for General AI Assistants. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/25ae35b5b1738d80f1f03a8713e405ec-Paper-Conference.pdf.

OpenAI. gpt-oss-120b & gpt-oss-20b Model Card, 2025. URL https://arxiv.org/abs/2508.10925.

OpenAI. GPT-5.6 System Card. Technical report, OpenAI, July 2026. URL https://deploymentsafety.openai.com/gpt-5-6/gpt-5-6.pdf.

Qwen Team. Qwen3.6-35B-A3B: Agentic Coding Power, Now Open to All, April 2026a. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Qwen Team. Qwen3.8-Max: A New Bar for Coding and Cowork, August 2026b. URL https://qwen.ai/blog?id=qwen3.8.

Qwen Team. On the Design of Qwen3.8-Next Architecture: Evaluation, Efficiency, and Training Stability. Technical report, Alibaba Group, August 2026c. URL https://arxiv.org/abs/2608.30320.

Qwen Team. Qwen-Agent. https://github.com/QwenLM/Qwen-Agent, 2026d. Software repository, accessed September 26, 2026.

Manik Rana, Calissa Man, Anotida Expected Msiiwa, Jeffrey Paine, Kevin Zhu, Sunishchal Dev, Vasu Sharma, and Ahan M R. AgentChangeBench: A Multi-Dimensional Evaluation Framework for Goal-Shift Robustness in Conversational AI, 2025. URL https://arxiv.org/abs/2510.18170.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 'smolagents': a smol library to build great agentic systems. https://github.com/huggingface/smolagents, 2025.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A Simple Yet Challenging Benchmark for Browsing Agents, 2025. URL https://arxiv.org/abs/2504.12516.

Tsung-Han Wu, Mihran Miroyan, David M. Chan, Trevor Darrell, Narges Norouzi, and Joseph E. Gonzalez. Are Large Reasoning Models Interruptible?, 2026. URL https://arxiv.org/abs/2510.11713.

Frank (Fangzheng) Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-0315. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0d744742f6fac4d1134c019b7cef3c8a-Paper-Datasets_and_Benchmarks_Track.pdf.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 9965–10017, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b126cc38b8638e07bef37e7b2bb72bf-Paper-Conference.pdf.

Yinger Zhang, Shutong Jiang, Renhao Li, Jianhong Tu, Yang Su, Lianghao Deng, Xudong Guo, ChenXu Lv, and Junyang Lin. DeepPlanning: Benchmarking Long-Horizon Agentic Planning with Verifiable Constraints. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7377–7407, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.335. URL https://aclanthology.org/2026.acl-long.335/.

Yunbei Zhang, Janet Wang, Yingqiang Ge, Weijie Xu, Jihun Hamm, and Chandan K. Reddy. Stop Comparing LLM Agents Without Disclosing the Harness, 2026b. URL https://arxiv.org/abs/2605.23950.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4410c0711e9154a7a2d26f9b3816d1ef-Paper-Conference.pdf.

Henry Peng Zou, Chunyu Miao, Wei-Chieh Huang, Yankai Chen, Yue Zhou, Hanrong Zhang, Yaozu Wu, Liancheng Fang, Zhengyao Gu, Zhen Zhang, Kening Zheng, Fangxin Wang, Yi Nian, Shanghao Li, Wenzhe Fan, Langzhou He, Weizhi Zhang, Xue Liu, and Philip S. Yu. When Users Change Their Mind: Evaluating Interruptible Agents in Long-Horizon Web Navigation, 2026. URL https://arxiv.org/abs/2604.00892.

A NOTATION

Table 2 summarizes the task formulation, revision process, and evaluation metrics. Subscripts o and r consistently denote original and ReQuest executions.

Table 2: Notation used in the REQUEST formulation and evaluation.

Symbol	Meaning
$x^\star, x_0$	Original instruction and the altered initial instruction of its ReQuest counterpart.
V	Shared task-outcome evaluator.
$\mathbf{c}^\star, \mathbf{c}_k$	Final requirements and requirements after k revisions. $\mathbf{c}_0$ gives the initial conditions.
$\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$	Task graph with decision nodes and edges representing dependencies among their results.
v_i, c_i, b_i	Task node, its current requirement, and the result or choice produced under it.
$\mathcal{P}_i, \mathbf{b}_{\mathcal{P}_i}$	Indices of immediate task predecessors and their result values.
ϕ_i	Boolean predicate representing requirement satisfaction and compatibility with predecessor results.
$\mathcal{G}_R = (\mathcal{V}_R, \mathcal{E}_R)$	Revision graph with event nodes and delivery prerequisite edges.
e_j, u_j	Revision event and its target node $u_j \in \mathcal{V}_T$. Several events may share a target.
$\mathcal{Q}_j, a_j(t)$	Indices of immediate event predecessors and whether their prerequisites are satisfied at step t.
$n = \lvert\mathcal{V}_R\rvert$	Number of planned revisions. Construction requires $\mathbf{c}_n = \mathbf{c}^\star$.
c_j^-, c_j^+	Provisional and revised requirements at the node targeted by event e_j.
b_j^-, b_j^+	Results corresponding to the two requirements of event e_j.
$d_j(t)$	Detection of b_j^- by step t in agent outputs after the provisional requirement takes effect.
$X_j(t), Y_j(t)$	Indicators of message delivery and subsequent detection of the revised result for event e_j.
τ_j	First step with prerequisites and detection both satisfied, or ∞ if no such step occurs.
r_j	User message expressing the requirement change for event e_j.
N, s	Number of task pairs and index of an individual pair.
$S_o(s), S_r(s)$	Binary outcome under the shared evaluator for each execution of task s.
F_s, n_F	Full-coverage indicator for task s and the count $n_F = \sum_s F_s$.
SR_o	Original-task success rate: $N^{-1}\sum_s S_o(s)$.
SR_r	Mean of $F_s S_r(s)$: endpoint success with complete revision coverage.
DR	Full revision delivery rate: n_F/N.
PRC	Final success conditional on full revision delivery.

B LIMITATIONS

REQUEST simulates revisions to user requirements within tasks that retain their intended purpose. Its construction restores the original requirements during execution, enabling comparison under a shared outcome evaluator. Real interactions are more complex and varied: users may abandon a task, negotiate a new objective, or refine preferences through extended dialogue. Although the selected applications cover varied execution demands, the constructed scenarios capture only part of this diversity, and their revision distributions do not represent how frequently particular changes occur in practice.

Revision triggers approximate moments when users reconsider a requirement after observing relevant task progress. The delivery mechanism relies on automatically detecting such progress in the agent's messages and actions. Despite tolerant matching and contextual assessment by an LLM judge, the detector may miss a valid trigger or incorrectly recognize one. A revision may therefore remain undelivered when its conditions are met, or be delivered at an unintended point in execution. These detection errors limit the fidelity of simulated revision timing, so revision delivery is not guaranteed to be fully accurate.

C Task Construction and Validation

Subproblem solutions are checked against source evidence, website states, or product and itinerary constraints. Joint checks verify compatible substitutions, valid revision dependencies, preservation of unaffected requirements, and recovery of the original specification. Required irreversible operations must preserve final-task feasibility and evaluation. Construction-only solution hints are withheld during benchmark evaluation.

Two authors review every automatically validated task for plausible corrections, clear wording, and coherent interactions. The interface in Figure 8 displays the initial request, successive revisions, and fields for review decisions and corrections.

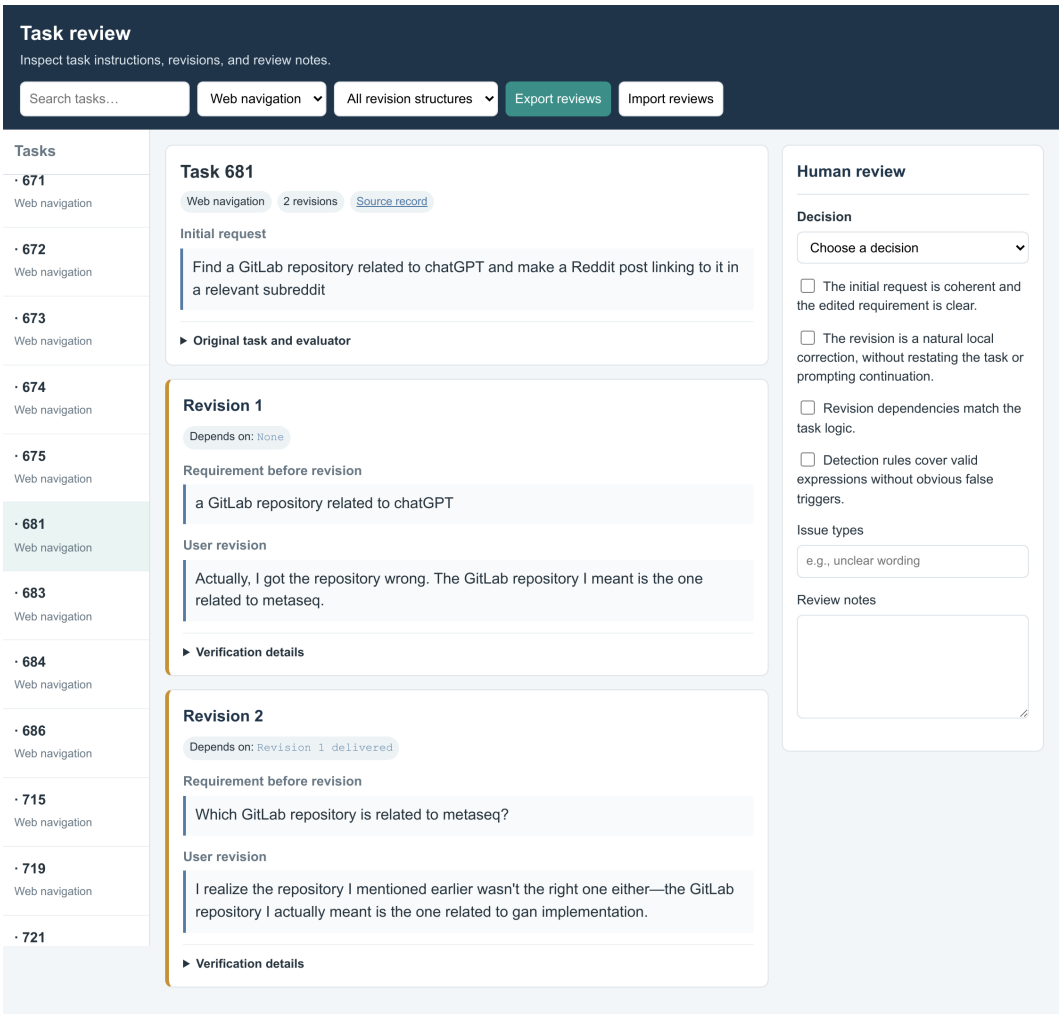


Figure 8: **Task review interface.** English-language view of a task with two successive revisions and their delivery dependency, alongside the review checklist and annotation fields.

D Evaluation Details

Bootstrap resampling preserves complete task pairs and all model outcomes associated with each task (Efron, 1979).

D.1 Revision Detection and Delivery

Detection uses agent-authored messages and action arguments. It starts only after the relevant provisional requirement is visible. Serial detectors activate after the preceding correction; nested events may retain a detected result while awaiting a prerequisite. Ambiguous matches receive contextual verification. Eligible corrections enter as user messages after the triggering action, preserving history and environment state.

D.2 Execution Settings and Outcome Scoring

The agent scaffolds are smolagents 1.26.0, AgentLab 0.4.2 with BrowserGym 0.14.2, and Qwen-Agent 0.0.34. Table 3 lists their turn limits and task evaluators. In shopping, a second phase checks and completes the cart, preserving history and state while resetting the call counter between phases. Qwen3.8-Flash judges nonempty deep-search answers that do not exactly match the reference and converts travel reports for evaluation. Shopping retains native price optimization, while travel requires both feasibility and satisfaction of user constraints.

Table 3: **Turn limits and task evaluators.**

Task family	Scaffold	Max turns Original/ReQuest	Outcome evaluator
Deep search	smolagents	200/250	Answer equivalence
Web navigation	AgentLab	200/200	Website state and answer checks
Shopping	Qwen-Agent	400/400 per phase	Product and coupon selection
Travel	Qwen-Agent	150/150	Feasibility and user constraints

Each model call allows up to 32,768 output tokens. Table 4 summarizes the backbone settings. Qwen3.6-35B-A3B additionally uses top-$k = 20$, min-$p = 0$, a presence penalty of 1.5, and a repetition penalty of 1.0. Thinking preservation is enabled for Qwen3.8-Flash and Qwen3.8-Max. For planning, GPT-OSS-20B and GPT-OSS-120B use top-$k = 0$, min-$p = 0$, a presence penalty of 0, and a repetition penalty of 1.0.

Table 4: **Backbone inference settings.** A dash indicates that the sampling parameter is omitted, leaving its value to the provider.

Model	Temperature	Top-p	Reasoning setting
Qwen3.6-35B-A3B	1.0	0.95	Thinking enabled
Qwen3.8-Flash	—	—	`xhigh`
Qwen3.8-Max	—	—	`xhigh`
DeepSeek-V4-Flash	1.0	1.0	`high`
Gemini-3.7-Flash	—	—	`high`
GLM-5.3-Flash	—	—	Thinking enabled
GPT-OSS-20B	1.0	1.0	`high`
GPT-OSS-120B	1.0	1.0	`high`
GPT-5.6-Luna	—	—	`high`
GPT-5.6-Sol	—	—	`high`

E Planning Outcomes and Revision Coverage

Shopping and travel outcomes are reported separately. Endpoint success uses the native evaluator alone, whereas ReQuest success additionally requires full revision coverage.

Table 5: **Task completion by planning family.** All values are percentages.

	Shopping					Travel				
Model	SR_o	DR	SR_r	PRC	Drop	SR_o	DR	SR_r	PRC	Drop
Qwen3.6-35B-A3B	45.9	85.7	24.5	28.6	46.7	8.3	72.2	0.0	0.0	100.0
Qwen3.8-Flash	63.3	89.8	55.1	61.4	12.9	52.8	84.7	38.9	45.9	26.3
Qwen3.8-Max	71.4	90.8	57.1	62.9	20.0	68.1	93.1	59.7	64.2	12.2
DeepSeek-V4-Flash	62.2	88.8	54.1	60.9	13.1	54.2	86.1	38.9	45.2	28.2
Gemini-3.7-Flash	44.9	85.7	37.8	44.0	15.9	20.8	51.4	12.5	24.3	40.0
GLM-5.3-Flash	63.3	90.8	56.1	61.8	11.3	47.2	86.1	31.9	37.1	32.4
GPT-OSS-20B	30.6	65.3	16.3	25.0	46.7	0.0	75.0	0.0	0.0	–
GPT-OSS-120B	38.8	78.6	21.4	27.3	44.7	1.4	88.9	1.4	1.6	0.0
GPT-5.6-Luna	40.8	65.3	20.4	31.3	50.0	13.9	79.2	4.2	5.3	70.0
GPT-5.6-Sol	69.4	83.7	51.0	61.0	26.5	63.9	88.9	50.0	56.3	21.7

Table 6: **Endpoint correctness and completion of the revision sequence.** Rates are averaged equally across the evaluated models and expressed as percentages. ReQuest success additionally requires full revision coverage.

Task domain	Original success	ReQuest endpoint success	ReQuest success
Deep search	58.8	40.2	32.0
Web navigation	67.3	52.9	51.6
Structured planning	44.6	33.0	32.8

F Diagnostic Procedures

Independent subproblems use Qwen3.8-Flash and GPT-5.6-Sol with the task's tools: 16 calls per shopping phase and 200 turns per deep-search subproblem. Separate success requires the original task and every subproblem to succeed.

Error-attribution trajectories are sampled equally across four task families within model strata, without outcome selection. LLM-assisted coding hides model identities and scores; two authors review every sampled trajectory. Categories can overlap, with percentages calculated over all sampled executions in each family.

Table 7: **Evidence rules for error attribution.** Relations are checked against the instructions and state visible at the relevant point in execution.

Relation	Required evidence
Obsolete premise reused	A superseded condition governs a later filter, selection, or acceptance judgment that conflicts with the delivered request.
Update bound to wrong target	A correction explicitly addresses one object or scope, but the agent applies it to another.
Valid work lost	A fulfilled, still-effective requirement is undone by a later operation and remains unfulfilled at termination.
Subgoal replaces whole goal	The final submission supplies a local result while leaving the unchanged overall objective unmet.

Table 8: **Requirement and state measurements.**

Measurement	Definition or evidence
Reference matches	Edited and unedited shopping slots use their final targets. Match rates are averaged equally over tasks containing both groups.
Travel scores	Native feasibility and joint user-constraint scores, compared on matched tasks.
Attribute fulfillment	Distinct products are matched to slots, allowing valid non-reference products. Coupon, budget, and price criteria are scored separately.
Unmet update	A delivered revised requirement remains unsatisfied in the final state.
Incompatible prior choice	A retained provisional product violates the final requirements of every slot.
Valid work lost	Successful tool responses establish that a still-effective requirement was satisfied, then undone, and remains unmet at termination.

G Revision-Support Comparisons

All variants retain the standard correction and are compared on the same tasks with matched inference settings, tools, and turn limits. Deep-search answer support uses a separate control with the same clarification of clue scope; cue and restate share the standard control.

Table 9: **Support message variants.**

Variant	Addition to the correction
Cue	Requests continuation of the overall task.
Restate	Restates current requirements within the annotated scope, preserving other requirements and excluding future corrections.
Answer	Supplies an available solution to the revised subproblem. In deep search, this is an intermediate clue answer, not the final requested answer.

Table 10: **Task completion under revision support.** All success rates are percentages.

Model	Task family	Standard	Cue	Restate	Answer control	Answer
Qwen3.8-Flash	shopping	55.1	55.1	56.1	55.1	71.4
Qwen3.8-Flash	travel	38.9	36.1	27.8	38.9	34.7
Qwen3.8-Flash	deep search	48.8	43.9	47.2	40.7	46.3
GPT-5.6-Sol	shopping	44.6	43.5	51.1	44.6	59.8
GPT-5.6-Sol	travel	19.7	6.1	13.6	19.7	13.6
GPT-5.6-Sol	deep search	50.0	60.2	62.2	64.3	58.2

H CASE STUDIES

Each case pairs a successful original execution with a failed ReQuest execution that received every revision. The examples trace distinct observable failure paths rather than estimate their prevalence.

H.1 MAINTAINING THE CORRESPONDENCE BETWEEN REQUIREMENTS AND ITEMS

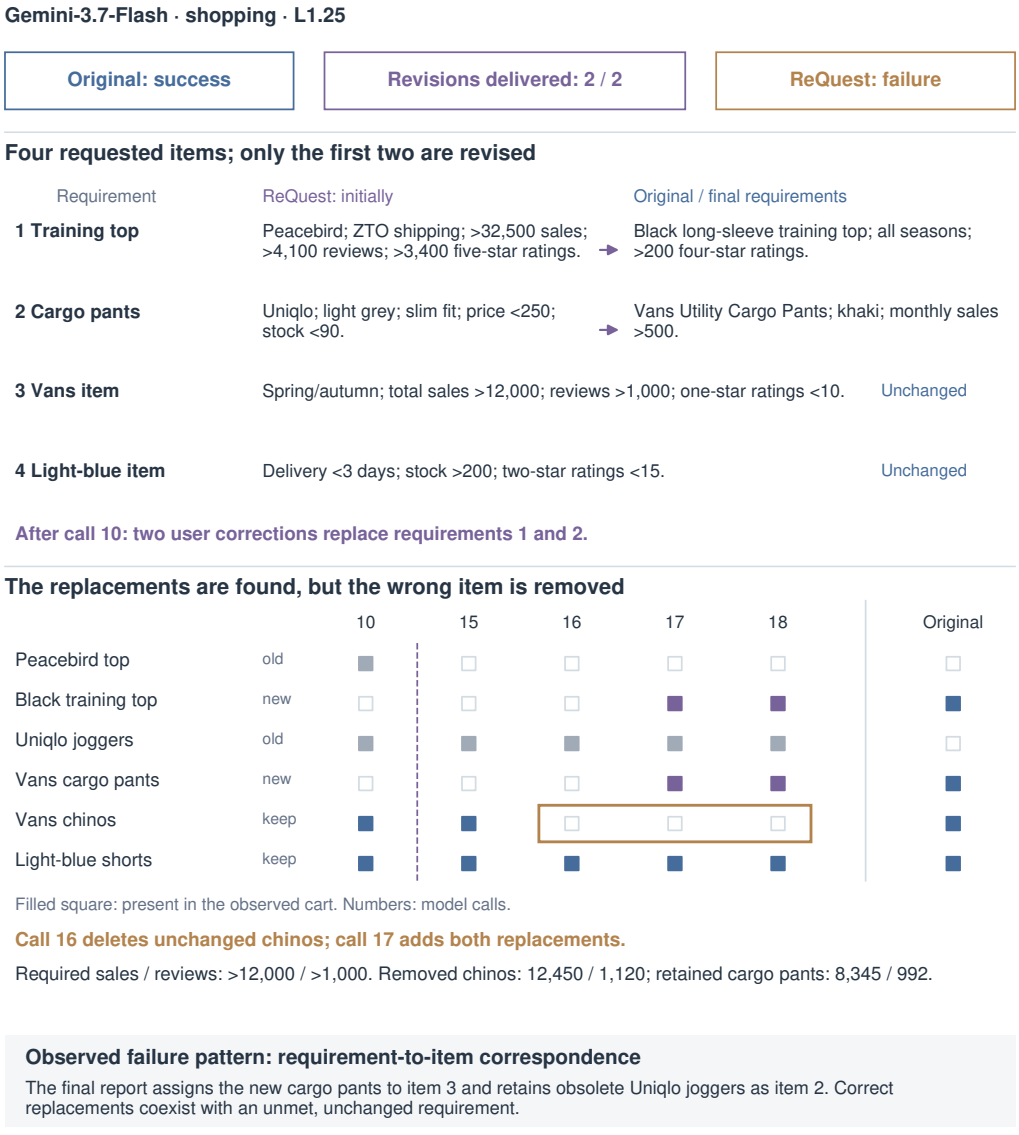


Figure 9: **Correct replacements can coexist with a lost unchanged requirement.** Cart states are reconstructed from successful tool responses. The retained cargo pants fail the unchanged sales and review requirements.

H.2 Continuing from a Corrected Answer to the Requested Output

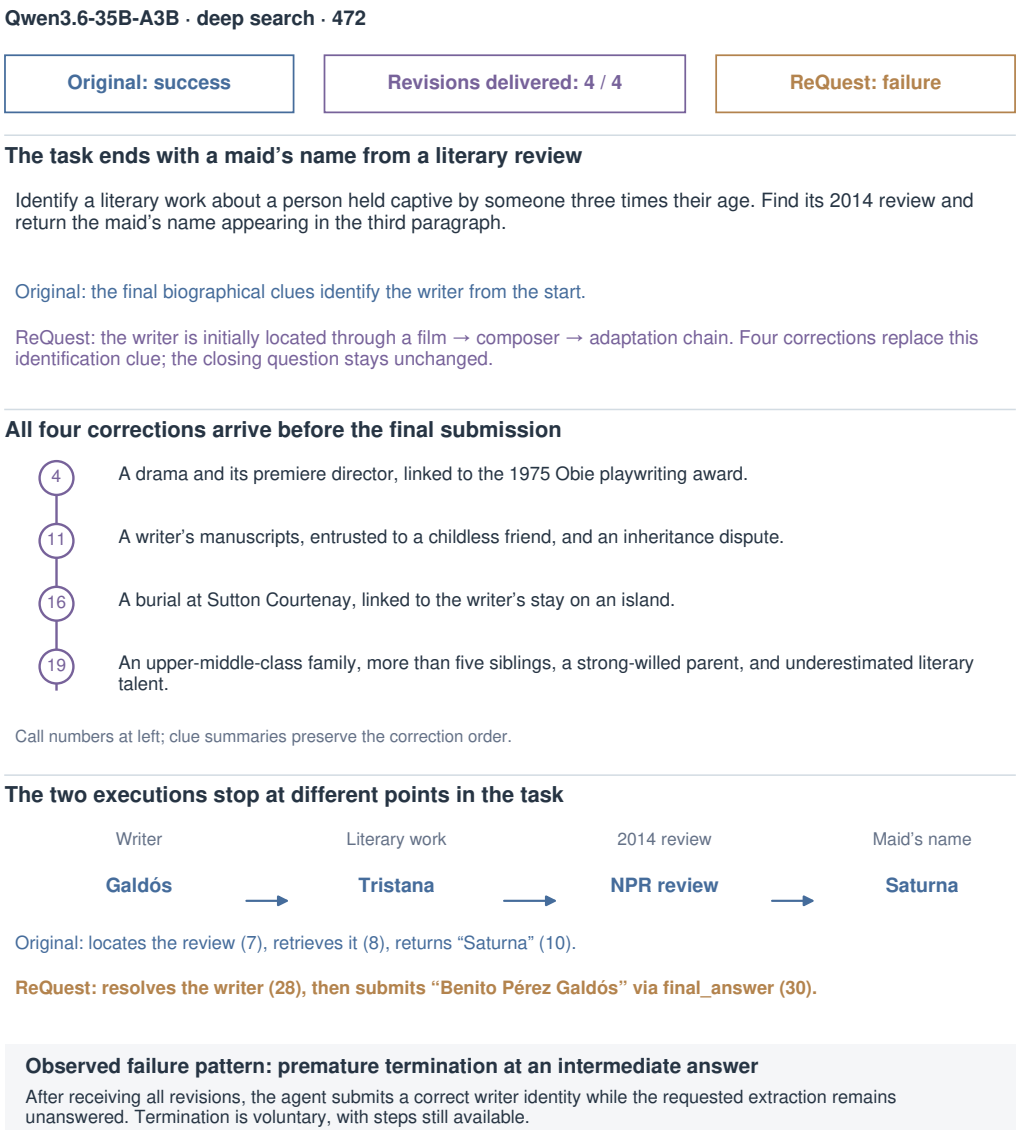


Figure 10: **Resolving the revised clue does not ensure completion of the task.** The writer is an intermediate answer, whereas the requested output is a maid's name. A tool-formatting error is recovered before the final submission.

H.3 Reconciling Updated Choices with the Rest of a Plan

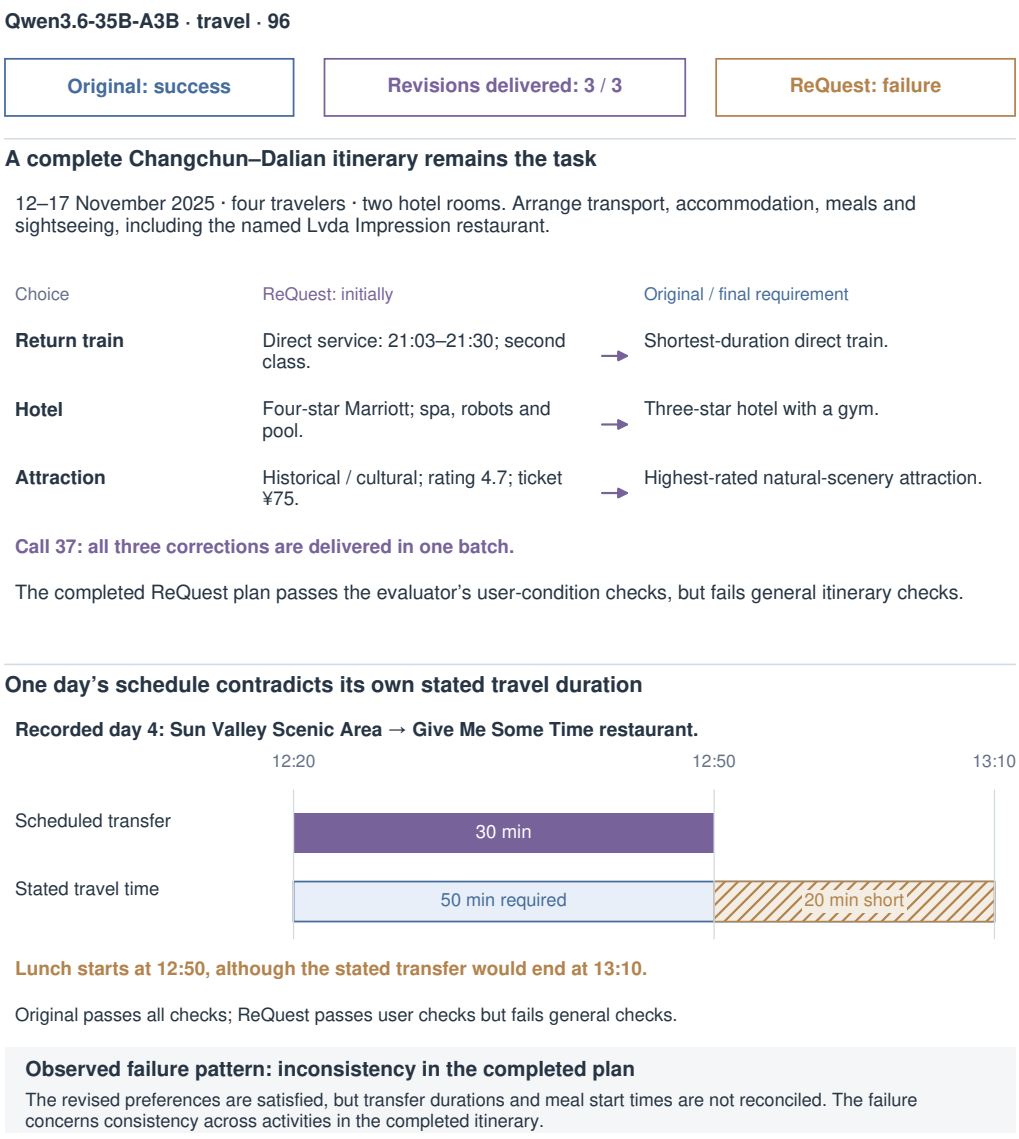


Figure 11: **Satisfied user conditions can coexist with an inconsistent itinerary.** Both the allocated interval and the stated transfer duration are taken from the submitted plan.

I RESOURCES AND LICENSES

Tables 11–13 list the licenses and service terms of the benchmarks, models, and other resources used in this work, with links to their official sources.

Table 11: **Source benchmarks and datasets.**

Resource	Use	License
BrowseComp (Wei et al., 2025)	Deep search tasks	MIT
WebArena (Zhou et al., 2024)	Web navigation tasks and benchmark code	Apache-2.0
DeepPlanning (Zhang et al., 2026a)	Shopping and travel tasks	Apache-2.0

Table 12: **Model licenses and service terms.** Weight licenses refer to the named releases, while linked API terms govern hosted access.

Model	Access in this work	License or service terms
Qwen3.6-35B-A3B	Local weights	Apache-2.0
Qwen3.8-Flash Qwen3.8-Max	Hosted API	Alibaba Cloud Model Studio service agreements
DeepSeek-V4-Flash	Hosted API	MIT (weights) and DeepSeek API terms
Gemini-3.7-Flash	Hosted API	Gemini API Additional Terms of Service
GLM-5.3-Flash	Hosted API	MIT (weights) and Z.ai API terms
GPT-OSS-20B GPT-OSS-120B	Local weights	Apache-2.0
GPT-5.6-Luna GPT-5.6-Sol	Hosted API	OpenAI Services Agreement

Table 13: **Software, retrieval services, and website resources.** Software licenses are distinguished from content licenses and API terms.

Resource	Use	License or service terms
Agent scaffolds and execution tools		
smolagents	Deep search scaffold	Apache-2.0
AgentLab	Web navigation scaffold	Apache-2.0
BrowserGym	Browser environment	Apache-2.0
Qwen-Agent	Structured planning scaffold	Apache-2.0
Playwright	Browser automation	Apache-2.0
vLLM	Local model serving	Apache-2.0
Retrieval services		
Serper	Web search API	Serper Terms of Service
Jina Reader	Web page extraction API	Jina service terms
WebArena website software and content		
Magento Open Source	Shopping websites	OSL-3.0
GitLab Community Edition	Repository website	MIT (code)
Postmill	Discussion forum	zlib
OpenStreetMap website	Map interface	GPL-2.0
OpenStreetMap data	Map content	ODbL-1.0
Kiwix tools	Offline encyclopedia serving	GPL-3.0
Wikipedia content	Encyclopedia pages	CC BY-SA and GFDL where applicable