InterruptBench: Evaluating LLM Agents Under User Interruptions in Long-Horizon Web Tasks

Anonymous ACL submission

Abstract

As LLM agents transition from short, static problem solving to executing complex, long-horizon tasks in dynamic environments, the ability to handle user interruptions, such as adding requirement or revising goals, during mid-task execution is becoming a core requirement for realistic deployment. However, existing benchmarks largely assume uninterrupted agent behavior or study interruptions only in short, unconstrained language tasks. In this paper, we present the first systematic study of interruptible agents in *long-horizon*, *environmentally grounded* web navigation tasks, where actions induce persistent state changes. We formalize three realistic interruption types, including addition, revision, and retraction, and introduce **InterruptBench**, a benchmark derived from WebArena-Lite that synthesizes high-quality interruption scenarios under strict semantic constraints. Using a unified interruption simulation framework, we evaluate six strong LLM backbones across single- and multi-turn interruption settings, analyzing both their effectiveness in adapting to updated intents and their efficiency in recovering from mid-task changes. Code and dataset are available at Anonymous Github Repository.

1 Introduction

The evolution of LLM agents from solving static, instantaneous problems to executing complex, multi-hour tasks in dynamic environments has inaugurated a new era of *realistic* agentic research (Durante et al., 2024; Wu et al., 2025c; Erdogan et al., 2025; Huang et al., 2026). Central to this realism are two critical dimensions: agents must be able to operate over a **long horizon**, handling hundreds of environmental interactions and issuing large amounts of actions, and must be **interruptible** to handle dynamic user intent, as users naturally want to examine partial results, correct spotted errors, or revise goals mid-execution over such a long horizon.

Despite the need for interruptible agents, existing research remains remarkably sparse. While recent benchmarks have advanced long-horizon capabilities (Liu et al., 2023; Chen et al., 2025a), they assume uninterrupted execution. The few works that do consider interruption focus on simple language tasks with severe limitations: they examine only short-horizon scenarios, lack grounded environmental constraints (Wu et al., 2025b), and restrict analysis to single-turn interruptions (Hua et al., 2024). Consequently, a critical gap remains in understanding **how agents can effectively and efficiently handle dynamic user interruptions in realistic, long-horizon settings** where environmental states are constrained, and actions have lasting consequences.

To address this gap, we present the first systematic study of interruptible agents in long-horizon, environmentally constrained settings. In this work, we focus on web navigation tasks, where agents must execute sequences of grounded actions (clicks, typing, navigation) that produce persistent state changes in dynamic web environments. We identify and formalize three realistic interruption scenarios that capture the spectrum of dynamic user intent: *addition* (adding new requirements to the initial query), *revision* (modifying an incorrect query to fix errors), and *retraction* (removing previously specified details).

To enable rigorous evaluation, we build upon high-quality, human-verified tasks from WebArena-Lite (Zhou et al., 2024; Koh et al., 2024; Wei et al., 2025) and develop a systematic synthesis pipeline to generate realistic interruption scenarios. For each original task, we synthesize an initial user query and subsequent user interruption messages, enforcing strict constraints to enhance data quality: (1) The initial user query combined with the user interruption messages should have the same meaning as the original intent and result in the same ground truth answers; (2) Each interruption mes-

sage should be essential, lacking any one interruption message should result in different ground truth answers than the original intent; (3) Each interruption message should be expressed in a natural and realistic user/customer tone. Our resulting benchmark, **InterruptBench**, spans three diverse interruption scenarios and five web domains, with each task requiring agents to handle user interruption messages during long-horizon mid-task execution.

We conduct extensive experiments across six powerful LLM backbones in WebAgent-R1 (Wei et al., 2025) agent scaffolding, including both open-source and large-scale closed-source models, evaluating their ability to handle interruptions across our three interruption scenarios. Our analysis examines not only how effectively agents adapt to updated intents following a user interruption but also how efficiently they do so. Beyond the single-turn interruption scenario, we extend our investigation to a multi-turn interruption scenario in which users issue multiple sequential interruptions during task execution, reflecting realistic interactive patterns over long horizons. We find that handling user interruptions effectively and efficiently during long-horizon agentic tasks remains challenging for powerful large-scale LLMs.

Our contributions are summarized as follows:

- We introduce **InterruptBench**, the first benchmark for studying how agents handle user interruptions in long-horizon, environmentally constrained settings, grounded in realistic web navigation tasks with persistent state changes.
- We develop a trajectory-grounded simulation and evaluation framework that injects interruptions at dynamically determined execution points and enables systematic measurement of both *effectiveness* and *efficiency* in post-interruption adaptation.
- We conduct extensive empirical analysis of LLM agents across six strong backbones under diverse user-interruption scenarios (addition, revision, retraction) and multi-turn interactions, revealing intriguing insights into their interruption-handling behavior.

2 InterruptBench: Benchmark Construction and Simulation

This section introduces the construction of our interruption benchmark and simulation. We first describe the underlying task suite and environment, then define three realistic mid-task interruption scenarios (Addition, Revision, and Retraction), and finally detail the procedure for simulating these interruptions during agent execution (Figure 1).

2.1 Task and Environment

We build our benchmark by augmenting tasks from WebArena-Lite (Zhou et al., 2024; Koh et al., 2024; Wei et al., 2025). WebArena-Lite provides realistic long-horizon web tasks with reliable constraints on environmental state and programmatic success checking. It covers five diverse domains and environments, including social forum (Reddit), collaborative coding (GitLab), e-commerce content management system (CMS), open street map (Map), and online shopping (OneStopShop), and provides a curated evaluation set of 165 human-verified tasks. Each task in WebArena-Lite requires the agent to interact with the environment via a sequence of observations and actions and can involve up to 30 LLM calls by default. More details about WebArena-Lite are provided in Appendix B.

2.2 Interruption Scenarios and Data Synthesis

In this work, we consider three realistic and most common interruption scenarios:

(1) Addition: An interruption scenario where a user initially provides a basic or incomplete query and then follows up with additional details, new information, or supplementary requirements to expand upon the original request.

(2) Revision: A modification interruption scenario where a user initially provides an incorrect or mistaken query and then follows up with corrections/modifications to fix the errors.

(3) Retraction: A retraction interruption scenario where a user initially provides a query with extra details/constraints and then follows up to remove or retract some of those details.

In addition, we consider a ***No-Interruption/No-Update*** scenario, where the agent receives only the initial user query, without any interruptions or additional follow-up during task execution.

Data Synthesis. For each original task in WebArena-Lite, we prompt a strong LLM to synthesize an initial user query and the additional user interruption messages to be injected during agent task execution. Notably, we enforce the following strict *constraints* to enhance data quality:

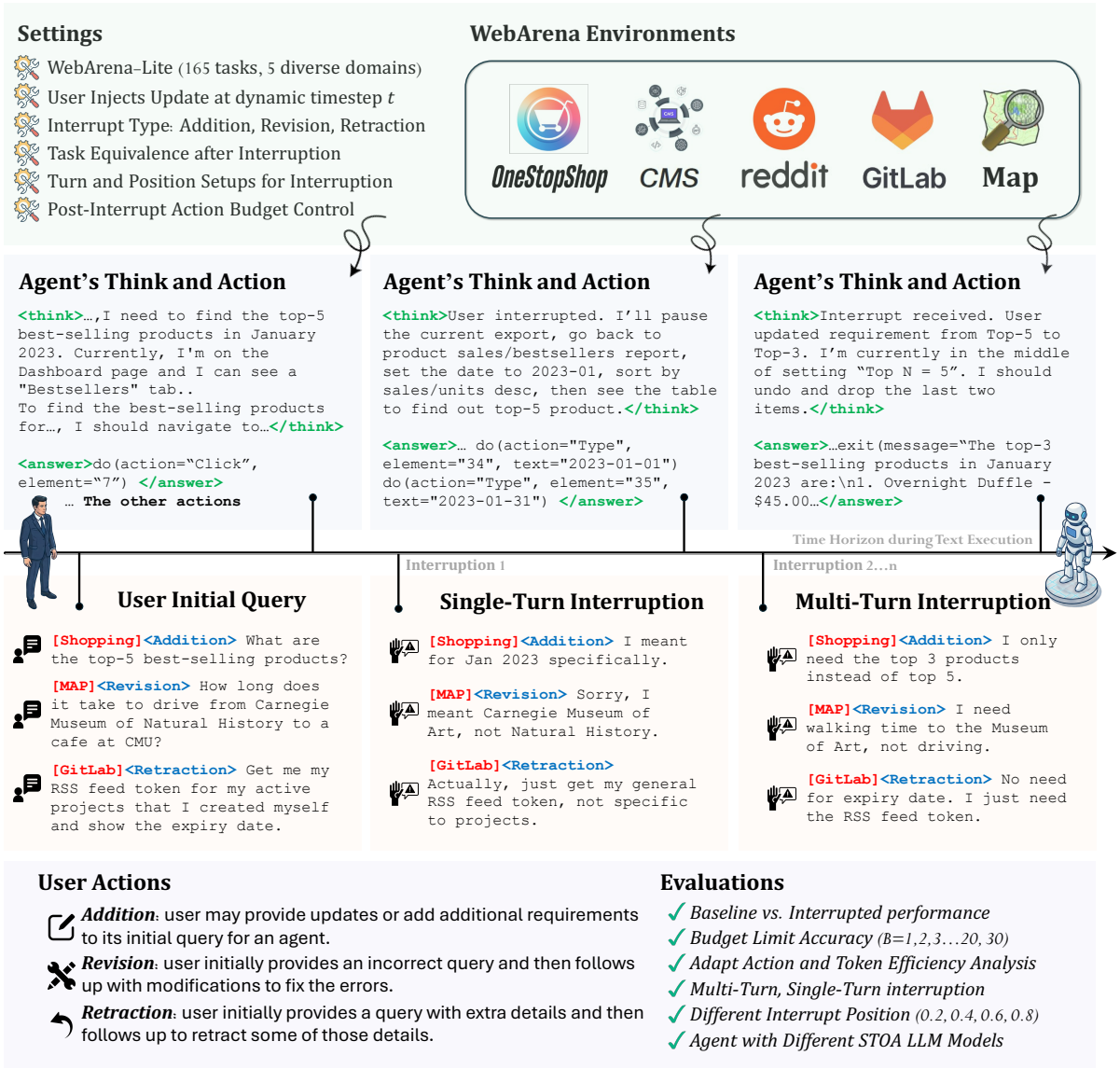


Figure 1: **InterruptBench setup and evaluation**. We evaluate an agent operating in the WebArena environment under user-driven interruptions at dynamic timesteps, where the user may update, revise, or retract parts of the original request.

(1) Ground Truth Consistency: The initial user query, combined with the user interruption messages, should have the same meaning as the original query in WebArena-Lite and lead to the same ground truth answer, enabling reliable evaluation.

(2) Necessity: Each interruption message should be essential; lacking any interruption message could result in a different ground truth answer from the original intent.

(3) Authenticity: Each interruption message should be expressed in a natural and realistic user/customer tone.

The complete prompts for synthesizing each type of interruption data are provided in Appendix G. To ensure data quality, each synthesized example is independently reviewed by three of the authors against the three criteria above; we report the full review protocol, inter-reviewer agreement statistics, and the relationship between Necessity and observed baseline success rates in Appendix C.

2.3 Trajectory-Grounded Interruption Simulation

We simulate user–agent interactions in which the agent first executes the initial user query and is then interrupted mid-task by additional user messages. Such simulation consists of three steps:

(1) Generate initial full trajectory. We first execute the initial user query to obtain a complete trajectory and record its total number of actions.

(2) Replay with progress-aware interruption. We then replay the trajectory sequentially and inject the interruption based on *relative progress* rather than a fixed global step. Specifically, for each task, the interruption is introduced at a predefined proportion of the initial trajectory, so the exact interruption step is automatically determined by the initial trajectory length and varies across tasks.

(3) Insert interruption and continue execution. Finally, we insert the additional user message at the selected interruption step and continue generating the post-interruption trajectory conditioned on the updated user intent.

This trajectory-grounded design enables controlled and comparable evaluation of interruption handling across tasks with varying lengths and complexities. We consider both *single-turn interruption* settings, in which only one interruption is introduced, and *multi-turn interruption* settings, in which users may interrupt the agent multiple times to provide iterative feedback.

3 Experimental Setup

3.1 Models

We evaluate six LLM backbones in the WebAgent-R1 (Wei et al., 2025) agent scaffolding: Claude-Haiku-4.5 (Anthropic, 2025b), Claude-Sonnet-4.5 (Anthropic, 2025c), Claude-Opus-4.5 (Anthropic, 2025a), Qwen3-Coder-480B-A35B (Yang et al., 2025) (abbreviated as Qwen3-480B-A35B), DeepSeek-V3.1 (Liu et al., 2024), and Mistral-Large-3 (Jiang et al., 2023). Unless noted, we keep the prompting format, tool/API surface, termination conditions, and maximum action budget identical across models. We leverage Claude-Opus-4.5 for data synthesis by default.

3.2 Interruption Configurations

Interruption scenarios and counts. We study the Addition, Revision, and Retraction interruption scenarios defined in Section 2.2. We evaluate both single-interruption episodes and multi-interruption episodes. For multi-interruption runs, we consider *random* sequences, where the interruption type can vary across turns. For example, an Addition followed by a Revision and then a Retraction. Throughout, interruptions are *informational*: they do not reset the agent or the environment, and do not invalidate progress already made. Instead, they provide additional user-supplied necessary information that the agent can leverage to finish the task more effectively.

Timing strategy. We inject interruptions at a relative position (60% by default) of the trajectory length obtained from the corresponding no-update run. To compare post-interruption adaptation across tasks with different pre-interruption lengths, we align trajectories using the post-interruption action index k, where $k{=}1$ is the first action after receiving an update. Evaluation under different interruption positions is provided in Appendix F.

3.3 Metrics

We evaluate agents along three complementary dimensions: task success, efficiency, and performance under multiple interruption updates.

Task success and post-interruption success curves. We define success with respect to the *final* post-update intent. We calculate the Success Rate (SR), which indicates whether a task is successfully completed according to three rule-based evaluation metrics provided in WebAgent-R1 (Wei et al., 2025): String Match, URL Match and Program Execution. To quantify how agents incorporate updated information over time, we report the post-interruption success curve, denoted as $\mathrm{SR}(k)$ (also referred to as the budget-limited success rate), defined as

$$\mathrm{SR}(k) \;=\; \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\big[\tau_i \le k\big], \qquad (1)$$

where τ_i denotes the number of actions taken after the interruption until success is achieved in episode i, and k is the post-interruption action index.

Efficiency. We measure output tokens and action count; output tokens serve as a proxy for the generation cost of producing actions and incorporating updated user intents. To isolate the effect of interruptions, we compare each interrupted run against a matched no-update run on the same underlying task. We additionally stratify episodes by paired outcomes to support fine-grained analysis.

Performance under multiple interruption updates. We report success as a function of the number of interruptions and study how $\mathrm{SR}(k)$ and representative failure patterns evolve as interruptions accumulate.

4 Experimental Results and Analysis

We structure our empirical analysis around three questions: **(Q1)** How do agents adapt to updated

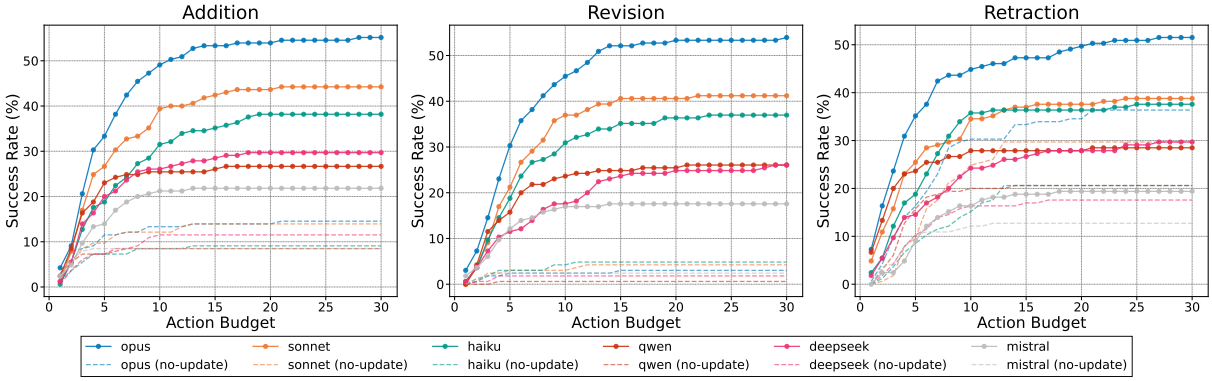


Figure 2: **Budget-limited success rate as a function of post-interruption action budget k across three interruption scenarios.** Solid lines denote runs that receive mid-task interruption updates, while dashed lines represent matched no-update runs that do *not* receive updates but are evaluated against the final intent, serving as a no-update reference under the same budget.

intents after an interruption? **(Q2)** How efficient is their adaptation? **(Q3)** How does performance change under multiple interruptions?

4.1 Adaptation Dynamics After Interruption

We analyze adaptation by aligning trajectories at the interruption point and computing the post-interruption success curve $\mathrm{SR}(k)$: the fraction of episodes that achieve the *final* intent within k actions after the interruption. This produces a family of curves $\mathrm{SR}_{m,t}(k)$ for model m and interruption type t. Figure 2 visualizes $\mathrm{SR}(k)$, where solid lines denote interrupted runs and dashed lines denote matched *no-update* runs that do not receive the interruption update but are still evaluated against the final intent.

Observed patterns. Figure 2 characterizes adaptation in the *budget-limited* setting, which operationalizes human-in-the-loop corrections under a constrained remaining action budget. Across all interruption types, success rate increases rapidly with budget up to approximately $k \approx 10$ and then plateaus, indicating that most adaptation occurs within the first few post-update actions. Model differences are apparent both in the rate of early improvement and in the eventual ceiling. For the Addition scenario, Opus reaches approximately 50% by $k = 10$ and 55% by $k = 30$, while Sonnet reaches roughly 40% and 44%, and Haiku reaches approximately 33% and 38%. Open-weight backbones plateau at lower levels; for instance, Qwen reaches approximately 26%, DeepSeek 30%, and Mistral 23% at $k = 30$.

The dashed no-update curves clarify how much of the observed performance depends on incorporating the update. For Revision, no-update curves remain near zero across all budgets for most models, indicating that success is largely contingent on correctly integrating the modified information; accordingly, the interrupted curves show substantial absolute gains (e.g., Opus exceeds 50% by $k \approx 15$). For Addition, no-update curves are higher but still well below the interrupted curves, suggesting that additions are partially compatible with the pre-update plan but still require explicit constraint integration. In contrast, Retraction exhibits notably stronger no-update performance, resulting in a smaller absolute gap between updated and no-update curves; here, the primary benefit of the update is earlier task completion under tight budgets rather than a substantially higher overall ceiling.

4.2 Efficiency of Adaptation

Success alone does not reveal whether an agent adapts efficiently. We therefore measure both token cost and action cost, and analyze whether models are truly reusing pre-interruption progress or effectively restarting from scratch.

Token and action efficiency. For each episode, we record output token usage T and action count A. We compare an interrupted run with update to a matched no-update run constructed from the same underlying original task. In Table 1, we primarily report absolute differences $\Delta_T = T_{\text{int}} - T_{\text{no-upd}}$ and $\Delta_A = A_{\text{int}} - A_{\text{no-upd}}$, so that positive values correspond to higher post-update cost.

Paired-outcome quadrants. To disentangle the effect of an update on *outcome* from its effect on *cost*, we assign each episode a paired label

Scenario: Addition								
Model	Metric	S/F	F/S	S/S	F/F	No-int	With-int	Avg.
Claude-Haiku-4.5	# cases	3.00	52.00	11.00	99.00	165.00	165.00	–
	# action	6.67	−2.31	−1.73	1.13	14.78	14.73	−0.04
	# token	1808.33	330.60	−187.64	2624.26	3896.01	5595.12	1699.11
Claude-Sonnet-4.5	# cases	4.00	55.00	18.00	88.00	165.00	165.00	–
	# action	21.75	−1.89	−3.89	2.53	13.24	14.06	0.82
	# token	8584.33	−123.77	−896.54	1128.42	3086.12	3756.99	670.87
Claude-Opus-4.5	# cases	1.00	70.00	21.00	73.00	165.00	165.00	–
	# action	0.00	−2.21	−1.86	0.41	11.09	10.10	−0.99
	# token	312.00	−92.70	159.29	350.53	1635.08	1773.00	137.92
Qwen3-235B-A22B	# cases	2.00	32.00	12.00	119.00	165.00	165.00	–
	# action	−2.00	−3.22	−2.92	1.73	13.12	13.50	0.39
	# token	−275.00	102.66	−218.00	51.08	298.48	336.04	37.56
DeepSeek-V3.1	# cases	5.00	35.00	14.00	111.00	165.00	165.00	–
	# action	6.40	−3.60	−3.50	1.51	14.27	14.42	0.15
	# token	1270.33	−238.92	−78.36	303.98	947.34	1133.00	185.66
Mistral-Large-3	# cases	5.00	27.00	9.00	124.00	165.00	165.00	–
	# action	1.40	0.56	−0.56	−0.81	11.45	10.95	−0.50
	# token	1115.80	165.14	6.00	36.41	2563.42	2651.95	88.53

Table 1: **Efficiency breakdown under the addition scenario.** We report action and token statistics grouped by paired outcomes between the no-interruption baseline and the interrupted run: S/F denotes baseline success and interruption failure; F/S denotes baseline failure and interruption success; S/S and F/F denote both succeed and both fail, respectively. **No-int** and **With-int** are averages over all episodes, and **Avg.** denotes the overall difference (With-int − No-int). Efficiency breakdowns for the revision and retraction scenarios are provided in Appendix D and Appendix E.

$(o_{\text{base}}, o_{\text{int}}) \in \{\text{S}, \text{F}\}^2$, where o_{base} and o_{int} denote success (S) or failure (F) of the matched no-update run and the interrupted run, respectively. This yields four exhaustive and mutually exclusive outcome quadrants:

- S/F (no-update succeeds, interrupted fails): the update is *outcome-degrading* for this episode.
- F/S (no-update fails, interrupted succeeds): the update is *outcome-improving* for this episode.
- S/S (both succeed): compare token/action costs under matched successful outcomes.
- F/F (both fail): analyze whether the update induces fail-fast behavior versus prolonged but unproductive interaction.

This breakdown reveals whether efficiency is coupled to success and whether models adapt by genuine reuse or implicit restarts.

Token overhead dominates adaptation cost. Table 1 shows that efficiency differences are driven primarily by *token* overhead rather than by additional actions. Across all models, the overall action delta is small in magnitude, whereas the token delta varies considerably. For example, under Addition, the average action overhead ranges from −0.99 (Opus) to +0.82 (Sonnet), while the average token overhead spans from +37.6 (Qwen) and +88.5 (Mistral) to +670.9 (Sonnet) and +1699.1 (Haiku). This suggests that incorporating an additional requirement frequently increases deliberation cost even when the number of actions remains comparable.

F/F failures incur the highest overhead. When conditioned on paired outcomes, the largest cost inflation occurs in the both-fail (F/F) quadrant: most models consume more actions and substantially more tokens before ultimately failing, consistent with prolonged but unproductive re-planning. This pattern is especially pronounced for Haiku (+2624 tokens in F/F) and Sonnet (+1128 tokens in F/F), and remains evident even for Opus (+351 tokens in F/F).

S/F cases are rare but costly. In the S/F quadrant (no-update success, interrupted failure), both action and token deltas are predominantly positive across most models (e.g., Sonnet: +21.75 actions and +8584 tokens; DeepSeek: +6.40 actions

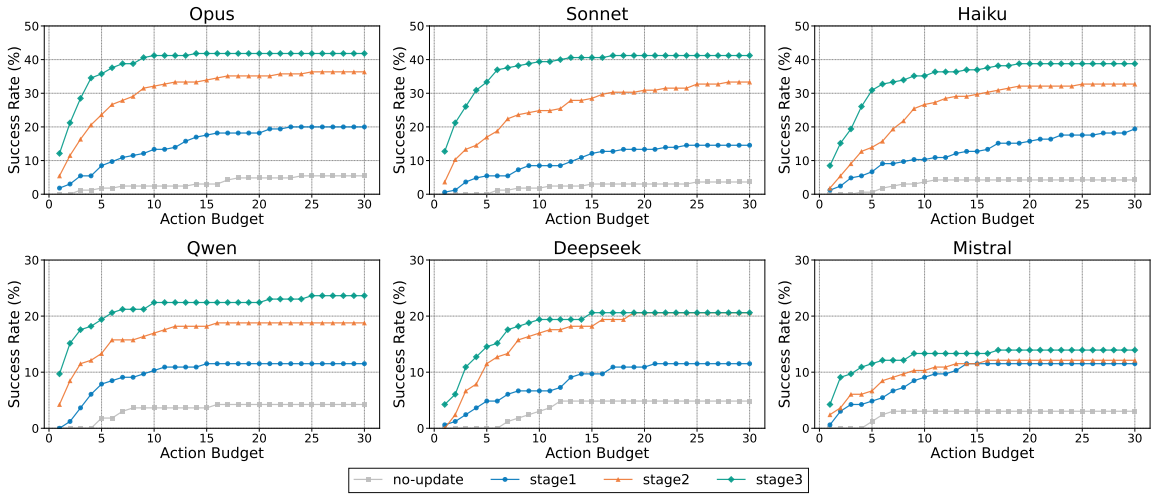


Figure 3: Post-update success curves $\mathrm{SR}_m(k \mid n)$ for $n \in \{1, 2, 3\}$ interruptions, aligned at the latest update (so k counts actions after the final interruption). Stage1/2/3 correspond to n=1/2/3. The figure shows how additional updates shift success upward under a fixed post-update budget, with diminishing returns as curves approach a model-specific plateau.

and +1270 tokens), indicating that when an interruption causes a previously successful task to fail, the agent tends to expend more effort before ultimately failing. Although S/F cases are relatively rare across all models (ranging from 1 to 5 cases), their existence reveals that interruptions can have a detrimental effect: rather than successfully integrating the update, the agent fails to reconcile the new information with its existing plan, resulting in task failure on episodes that would have succeeded without the interruption.

F/S outcomes highlight model capability gaps. The F/S quadrant (no-update failure, interrupted success) is the most prevalent outcome category after F/F, and reveals substantial variation across models. Stronger models show more F/S recoveries (Opus: 70; Sonnet: 55), while open-weight models exhibit fewer recovered cases (DeepSeek: 35; Qwen: 32; Mistral: 27). Efficiency also differs within recovered episodes: weaker models require extra overhead (e.g., Haiku: +6.67 actions and +1808 tokens), whereas stronger models can recover with near-zero or negative overhead (Opus: −2.21 actions and −92.7 tokens).

S/S adaptation is typically lightweight. Finally, in the both-succeed (S/S) quadrant, both Δ_A and Δ_T are comparatively small across models, indicating that successful adaptation often reuses pre-interruption progress without requiring a full restart. Among the strongest models, this manifests as near-zero or negative action overhead, reflecting more direct task completion after integrating the update.

Model	0 int.	1 int.	2 int.	3 int.
Claude-Haiku-4.5	5.45	19.39	33.33	38.79
Claude-Sonnet-4.5	4.24	15.15	33.33	41.21
Claude-Opus-4.5	5.45	21.21	36.36	41.82
Qwen3-235B-A22B	4.24	11.52	18.79	23.64
DeepSeek-V3.1	4.85	12.12	21.21	20.61
Mistral-Large-3	3.03	11.52	12.73	13.94

Table 2: Converged performance under multiple interruption updates. Success rates under $n \in \{0, 1, 2, 3\}$ interruptions (more interruptions correspond to more user-provided information).

4.3 Performance Under Multiple Informational Interruptions

This section evaluates agent performance in episodes with $n \in \{1, 2, 3\}$ interruptions, where each interruption provides an additional user message containing necessary task information. The agent continues from the same environment state and trajectory; each update only augments the information available to the model without resetting any prior progress. We examine how success rate evolves as information is progressively supplied (Figure 3), quantify the marginal gain in success as n increases, assess whether improvements remain stable as updates accumulate, and compare how different LLM backbones leverage multiple interruption updates (Table 2).

Additional user interruption updates generally improve success. Across all models, Figure 3 and

Table 2 shows that providing more information through additional interruptions generally leads to higher success rates. This supports the premise that incremental clarification or details can help an agent complete long-horizon web tasks without resetting or derailing its ongoing progress.

Claude models improve more consistently and strongly. Models differ substantially in how predictably and effectively they exploit accumulating information. The Claude model family exhibits consistent and monotonic improvements across all stages, with Opus achieving the highest overall performance (41.82% at $n = 3$) and Sonnet showing the largest marginal gain between $n = 1$ and $n = 2$ (an increase of over 18 percentage points). Notably, all three Claude models roughly double their success rate from $n = 1$ to $n = 3$, indicating a robust capacity to reliably integrate successive updates and translate them into measurable performance gains.

Open-weight models gain less and are less stable. Open-weight models also benefit from additional interruptions, but exhibit smaller marginal gains and less stable improvement patterns. Among them, Qwen shows gradual but consistent growth, roughly doubling its success rate from $n = 1$ to $n = 3$, while Mistral improves only marginally across stages, suggesting a limited capacity to exploit progressively revealed task details. DeepSeek-V3.1 presents a particularly instructive case: it achieves a substantial gain from $n = 1$ to $n = 2$ but slightly regresses at $n = 3$, illustrating a failure mode in which later updates are imperfectly integrated and can interfere with previously incorporated information.

5 Case Study

In this section, we present a representative case study from the single-interruption *Addition* setting in the WebArena Map environment. As illustrated in Figure G.1, the agent begins with the original goal of comparing walking versus driving time to CMU, takes early steps in the web UI such as opening directions, filling the start location, and clicking "Go," and then receives an interrupt that adds a new constraint: the origin should be Randyland. This update is not a minor wording change; it changes the task's execution state and therefore the ground-truth comparison. The example is intended to illustrate, at the trajectory level, how successful and unsuccessful interruption handling differ in web interaction.

The contrast is that the failure trace treats the interrupt as a surface-level message: it may acknowledge the update, but it does not fully override the earlier assumption in the environment (the "From" field and computed route remains tied to the old origin), so the final reported comparison is conditioned on stale state. In the correct trace, the agent treats the interrupt as a state-changing event: it explicitly detects intent drift, retracts the previous origin assumption, edits the "From" field to Randyland, triggers a recomputation, and only then reports the updated walking–driving difference. Overall, the case study highlights a key requirement for interruptible agents: effective handling is not just updating the final answer, but ensuring the UI state and intermediate computations are reconciled with the post-interrupt intent before producing any result.

6 Conclusion

We present InterruptBench, a benchmark for evaluating LLM agents under user interruptions in long-horizon, environmentally constrained web tasks. By formalizing addition, revision, and retraction interruptions, our framework enables controlled comparisons between no-update and interrupted executions while measuring both post-interruption success and adaptation efficiency. Across six strong backbones, we find that agents still struggle to reconcile updated user intent with persistent environment state, often carrying forward stale assumptions or incurring extra recovery cost. These findings highlight interruption handling as a key bottleneck for realistic agent deployment and point to the need for stronger state tracking, intent reconciliation, and error recovery mechanisms.

Limitations

Our benchmark focuses on WebArena-Lite web tasks, so the findings may not fully generalize to other long-horizon domains such as software engineering, robotics, or multimodal desktop workflows. The interruption scenarios are synthesized and manually validated, but they may not capture the full diversity, ambiguity, and timing of real user interruptions. Future work can extend InterruptBench to broader environments and collect naturally occurring human-agent interruption traces.

References

Anthropic. 2025a. Claude opus 4.5 system card.

Anthropic. 2025b. System card: Claude haiku 4.5.

Anthropic. 2025c. System card: Claude sonnet 4.5.

Ruhana Azam, Aditya Vempaty, and Ashish Jagmohan. 2025. Reflection-based memory for web navigation agents. *Preprint*, arXiv:2506.02158.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*.

Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, and 1 others. 2025a. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations. *arXiv preprint arXiv:2506.13651*.

Yankai Chen, Xinni Zhang, Yifei Zhang, Yangning Li, Henry Peng Zou, Chunyu Miao, Weizhi Zhang, Xue Liu, and Philip S. Yu. 2025b. Embracing trustworthy brain-agent collaboration as paradigm extension for intelligent assistive technologies. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems Position Paper Track*.

Mahir Labib Dihan, Tanzima Hashem, Mohammed Eunus Ali, and Md Rizwan Parvez. 2025. Weboperator: Action-aware tree search for autonomous agents in web environment. *Preprint*, arXiv:2512.12692.

Zane Durante, Qiuyuan Huang, Naoki Wake, Ran Gong, Jae Sung Park, Bidipta Sarkar, Rohan Taori, Yusuke Noda, Demetri Terzopoulos, Yejin Choi, and 1 others. 2024. Agent ai: Surveying the horizons of multimodal interaction. *arXiv preprint arXiv:2401.03568*.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-act: Improving planning of agents for long-horizon tasks. *arXiv preprint arXiv:2503.09572*.

Xueyang Feng, Zhi-Yuan Chen, Yujia Qin, Yankai Lin, Xu Chen, Zhiyuan Liu, and Ji-Rong Wen. 2024. Large language model-based human-agent collaboration for complex task solving. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1336–1357.

Tianyu Fu, Anyang Su, Chenxu Zhao, Hanning Wang, Minghui Wu, Zhe Yu, Fei Hu, Mingjia Shi, Wei Dong, Jiayao Wang, Yuyang Chen, Ruiyang Yu, Siran Peng, Menglin Li, Nan Huang, Haitian Wei, Jiawei Yu, Yi Xin, Xilin Zhao, and 4 others. 2025. Mano technical report. *Preprint*, arXiv:2509.17336.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2025. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations (ICLR)*. Oral Presentation.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, Bangkok, Thailand. Association for Computational Linguistics.

Wenyue Hua, Mengting Wan, Shashank Vadrevu, Ryan Nadel, Yongfeng Zhang, and Chi Wang. 2024. Interactive speculative planning: Enhance agent efficiency through co-design of system and user interface. *arXiv preprint arXiv:2410.00079*.

Wei-Chieh Huang, Weizhi Zhang, Yueqing Liang, Yuanchen Bei, Yankai Chen, Tao Feng, Xinyu Pan, Zhen Tan, Yu Wang, Tianxin Wei, and 1 others. 2026. Rethinking memory mechanisms of foundation agents in the second half. *arXiv preprint arXiv:2602.06052*.

Dongsheng Jiang, Yuchen Liu, Songlin Liu, Jin'e Zhao, Hao Zhang, Zhen Gao, Xiaopeng Zhang, Jin Li, and Hongkai Xiong. 2023. From clip to dino: Visual encoders shout in multi-modal large language models. *arXiv preprint arXiv:2310.08825*.

Imene Kerboua, Sahar Omidi Shayegan, Megh Thakkar, Xing Han Lù, Léo Boisvert, Massimo Caccia, Jérémy Espinas, Alexandre Aussem, Véronique Eglin, and Alexandre Lacoste. 2025. Focusagent: Simple yet effective ways of trimming the large context of web agents. *Preprint*, arXiv:2510.03204.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905, Bangkok, Thailand. Association for Computational Linguistics.

Ziyue Li, Yuan Chang, Gaihong Yu, and Xiaoqiu Le. 2025. Hiplan: Hierarchical planning for llm-based agents with adaptive global-local guidance. *Preprint*, arXiv:2508.19076.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. 2024. Omniparser for pure vision based gui agent. *Preprint*, arXiv:2408.00203.

Chunyu Miao, Henry Peng Zou, Yangning Li, Yankai Chen, Yibo Wang, Fangxin Wang, Yifan Li, Wooseong Yang, Bowei He, Xinni Zhang, Dianzhi Yu, Hanchen Yang, Hoang H Nguyen, Yue Zhou, Jie Yang, Jizhou Guo, Wenzhe Fan, Chin-Yuan Yeh, Panpan Meng, and 10 others. 2026. RECODE: A benchmark for research code DEvelopment with interactive human feedback. In *The Fourteenth International Conference on Learning Representations*.

OpenAGI Foundation. 2025. Lux: The state-of-the-art foundation computer-use model. Technical report, OpenAGI Foundation.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. 2025. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *Preprint*, arXiv:2411.02337.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Zhiwei Liu, Jianguo Zhang, Haolin Chen, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, and 1 others. 2025. Userbench: An interactive gym environment for user-centric agents. *arXiv preprint arXiv:2507.22034*.

Yijia Shao, Vinay Samuel, Yucheng Jiang, John Yang, and Diyi Yang. 2024. Collaborative gym: A framework for enabling and evaluating human-agent collaboration. *arXiv preprint arXiv:2412.15701*.

Yijia Shao, Humishka Zope, Yucheng Jiang, Jiaxin Pei, David Nguyen, Erik Brynjolfsson, and Diyi Yang. 2025. Future of work with ai agents: Auditing automation and augmentation potential across the us workforce. *arXiv preprint arXiv:2506.06576*.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, and 1 others. 2025. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*.

Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. 2025a. Collabllm: From passive responders to active collaborators. In *Forty-second International Conference on Machine Learning*.

Tsung-Han Wu, Mihran Miroyan, David M Chan, Trevor Darrell, Narges Norouzi, and Joseph E Gonzalez. 2025b. Are large reasoning models interruptible? *arXiv preprint arXiv:2510.11713*.

Yaozu Wu, Dongyuan Li, Yankai Chen, Renhe Jiang, Henry Peng Zou, Wei-Chieh Huang, Yangning Li, Liancheng Fang, Zhen Wang, and Philip S Yu. 2025c. Multi-agent autonomous driving systems with large language models: A survey of recent advances, resources, and future directions. *Findings of the Association for Computational Linguistics: EMNLP*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*. Datasets and Benchmarks Track.

Taofeng Xue, Chong Peng, Mianqiu Huang, Linsen Guo, Tiancheng Han, Haozhe Wang, Jianing Wang, Xiaocheng Zhang, Xin Yang, Dengchang Zhao, Jinrui Ding, Xiandi Ma, Yuchen Xie, Peng Pei, Xunliang Cai, and Xipeng Qiu. 2026. Evocua: Evolving computer use agents via learning from scalable synthetic experience. *Preprint*, arXiv:2601.15876.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2024. Agentoccam: A simple yet strong baseline for llm-based web agents. *Preprint*, arXiv:2410.13825.

Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. 2024. Exact: Teaching ai agents to explore with reflective-mcts and exploratory learning. *Preprint*, arXiv:2410.02052.

Michael JQ Zhang, W Bradley Knox, and Eunsol Choi. 2024. Modeling future conversation turns to teach llms to ask clarifying questions. In *The Thirteenth International Conference on Learning Representations*.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. GPT-4V(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning (ICML)*.

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. 2024. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations (ICLR)*.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. 2025. Scaling test-time compute for llm agents. *Preprint*, arXiv:2506.12928.

Henry Peng Zou, Wei-Chieh Huang, Yaozu Wu, Yankai Chen, Chunyu Miao, Hoang Nguyen, Yue Zhou, Weizhi Zhang, Liancheng Fang, Langzhou He, and 1 others. 2025a. A survey on large language model based human-agent systems. *Authorea Preprints*.

Henry Peng Zou, Wei-Chieh Huang, Yaozu Wu, Chunyu Miao, Dongyuan Li, Aiwei Liu, Yue Zhou, Yankai Chen, Weizhi Zhang, Yangning Li, and 1 others. 2025b. A call for collaborative intelligence: Why human-agent systems should precede ai autonomy. *arXiv preprint arXiv:2506.09420*.

A Related Work

A.1 Web-Based Autonomous Agents

Recent work on web agents shows a consistent evolution across perception, planning, learning, and evaluation. To overcome the limitations of raw HTML/DOM representations, prior studies either adopt vision-centric grounding to localize actions directly from screen pixels (Lu et al., 2024; Gou et al., 2025; Zheng et al., 2024), or reduce observation complexity by pruning DOM or accessibility trees to retain task-relevant information (Yang et al., 2024; Kerboua et al., 2025). For long-horizon tasks, flat ReAct-style loops have been replaced by more structured reasoning mechanisms, including tree search and test-time compute scaling (Dihan et al., 2025; Yu et al., 2024; Zhu et al., 2025), as well as hierarchical planning and reflection-based control (Li et al., 2025; Azam et al., 2025). Meanwhile, training paradigms have shifted from pure supervised fine-tuning toward environment-feedback-driven approaches, such as curriculum reinforcement learning and end-to-end multi-turn optimization (Qi et al., 2025; Wei et al., 2025; Xue et al., 2026). Recent computer-use foundation models demonstrate impressive capabilities in autonomous desktop and browser control (OpenAGI Foundation, 2025; Fu et al., 2025). Correspondingly, evaluation has moved from static webpages to dynamic, containerized benchmarks, established by (Zhou et al., 2024) and extended to multimodal understanding in realistic web interactions (He et al., 2024; Koh et al., 2024) and end-to-end computer- or operating-system-level control (Xie et al., 2024). As agents tackle longer web tasks, user interaction becomes more dynamic, with mid-task updates like additions, revisions, or retractions. Yet most benchmarks assume fixed user intent and uninterrupted execution, leaving agent behavior under realistic interruptions underexplored. Our paper introduces a benchmark and analysis framework that explicitly evaluates how web agents adapt and recover under such user interruptions to bridge this gap.

A.2 Human Agent Collaboration System

Human-agent collaboration systems aim to combine human judgment and domain expertise with the scalability and automation of agents, improving task performance, reliability, and safety in real-world applications (Zou et al., 2025a,b; Shao et al., 2025). These systems treat humans as integral components that provide additional information (Qian et al., 2025), feedback (Miao et al., 2026), or control (Barres et al., 2025; Chen et al., 2025b) to compensate for the limitations of fully autonomous agents and to improve reliability and safety. Early work in this area primarily treats humans as reactive helpers, with the agent deciding when and how to request help. Zhang et al. (2024) train LLMs to predict future conversation turns and to proactively ask clarifying questions when an instruction is ambiguous, thereby improving downstream task performance while keeping humans in a primarily response role. CollabLLM (Wu et al., 2025a) similarly fine-tunes models to move from passive responders to active collaborators that generate clarifying questions and suggestions over multi-turn dialogue, but the user still mainly reacts to model-initiated queries. ReHAC (Feng et al., 2024) further introduces a reinforcement-learning policy that chooses opportune stages for human intervention during complex, multi-step tasks, demonstrating gains from limited, well-timed human input but under fixed task goals and relatively short horizons. More recent work shifts toward settings where humans proactively steer, coordinate with, and intervene in agents, rather than being confined to purely reactive assistance. Collaborative Gym (Shao et al., 2024) proposes an open framework where humans and agents engage in bidirectional, non-turn-taking communication while interacting with task environments (e.g., travel planning, related-work writing, tabular analysis), and evaluates both outcome and process metrics. Wu et al. (2025b) study how large reasoning models behave when users may interrupt them mid-reasoning or update the problem specification in flight. They reveal severe performance drops and distinct failure modes, such as reasoning leakage, panic, and self-doubt under these user-like interventions.

However, these studies mostly focus on language-only or abstract communication settings and usually treat the task as a fixed, static prob-

lem. As a result, they do not capture long-horizon agents acting in persistent environments, where actions update external state and users may change their minds mid-task through additions, revisions, or retractions. In this paper, we treat such user interruptions as a core, realistic form of human–agent collaboration for long-horizon web agents, and introduce a benchmark and analysis framework that jointly evaluates their success and efficiency under these environment-bound interruptions.

B Web Environment & Dataset Statistics

WebArena-Lite. WebArena (Zhou et al., 2024) is a realistic, self-hostable web environment for developing LLM-based agents. It comprises 812 real-world web tasks spanning diverse domains, including social forum (Reddit), collaborative coding (GitLab), e-commerce content management system (CMS), open street map (Map), and online shopping (OneStopShop). WebArena-Lite (Koh et al., 2024; Wei et al., 2025) is a curated version of WebArena designed for more reliable evaluation. It selects 165 representative tasks for human verification as the evaluation set and uses the remaining 647 tasks for training. For each website, the authors summarize the core functionalities and valid items and construct a set of task prototypes and manually implement rule-based solvers using Playwright scripts for each prototype. The corresponding solvers are executed on the websites to collect ground-truth trajectories. For each of the 165 tasks in WebArena-Lite, we synthesized three interruption data corresponding to each interruption scenario (addition, revision, and retraction), using prompts provided in Appendix G.

C Additional Details on Human Validation of InterruptBench

Human review protocol. After LLM-based synthesis, all interruption examples were manually reviewed before being included in InterruptBench. The review focused on three criteria that directly correspond to the constraints in Section 2.2: (1) *ground-truth consistency*, i.e., the initial query together with the interruption message recovers the same user intent and ground-truth answer as the original WebArena-Lite task; (2) *necessity*, i.e., the interruption message changes or completes task-relevant information such that omitting it would generally lead to a different target intent; and (3) *authenticity*, i.e., the interruption is phrased as a natural mid-task user update rather than as an artificial benchmark instruction.

To quantify the reliability of this review, we additionally conducted an independent human audit on a stratified subset of 120 examples, with 40 examples from each interruption type: Addition, Revision, and Retraction. Each example was independently reviewed by three of the authors using binary labels for the three criteria above. The majority-pass rate across all three criteria was 99.2%, indicating that almost all audited examples satisfied the intended construction constraints under majority judgment. At the criterion level, unanimous positive agreement was 96.7% for ground-truth consistency, 89.2% for necessity, and 90.0% for authenticity. These results suggest that the synthesized interruptions are usually judged as semantically valid, task-relevant, and natural by multiple human reviewers.

Clarifying the no-interruption baseline. The no-interruption baseline in Figure 2 should not be interpreted as a setting in which success is mathematically impossible. Instead, it serves as a lower-bound reference in which the agent receives only the transformed initial query but is still evaluated against the final post-update intent. Therefore, a non-zero success rate is expected in some cases. For example, an underspecified initial query may still lead the agent to select a default value that happens to match the final intent; a revised or retracted constraint may not affect every intermediate action before the final answer; or the environment/evaluator may accept an answer that overlaps with the final target even without the explicit update. This is especially common for Retraction, where the initial query is often more constrained than the final query, so executing the more specific request can sometimes still satisfy the less specific post-retraction intent.

Thus, the construction constraint is not that the initial query must always cause agent failure. Rather, the intended constraint is semantic: the interruption should provide task-relevant information that changes, corrects, or relaxes the user intent relative to the initial query. The dashed curves in Figure 2 therefore measure how often agents can accidentally or partially satisfy the final intent without receiving the update, while the gap between dashed and solid curves measures the benefit of explicitly incorporating the interruption. This interpretation also explains why the no-interruption baseline is

near zero for many Revision cases, higher for some Addition cases, and highest for Retraction cases.

D Efficiency Analysis under Revision Interruptions

Table D.1 summarizes paired outcomes and efficiency statistics under the scenario shown above. Across models, outcome disagreements are predominantly concentrated in F/S (interruption failure, no-update success) rather than S/F (interruption success, no-update failure); for instance, Claude-Opus exhibits F/S$= 86$ versus S/F$= 1$, and Claude-Sonnet shows F/S$= 62$ versus S/F$= 1$. With respect to efficiency, interruptions reduce the average number of actions for several models (negative **Avg.** for number action on the Claude family and DeepSeek), and are often accompanied by token reductions (negative **Avg.** for number of token). This trend is not universal: Qwen shows a slight increase in actions despite fewer tokens, while Mistral incurs substantial additional token usage. These results indicate that the efficiency impact of interruption handling is highly model- and scenario-dependent, and can entail trade-offs between interaction length and generation cost.

E Efficiency Analysis under Retraction Interruptions

Relative to Table D.1, Table E.2 places more mass on S/S and exhibits non-trivial S/F counts, suggesting that interruptions can correct a subset of no-update failures in this scenario (while still introducing some F/S cases). Notably, all models exhibit lower average action counts with interruptions (negative **Avg.** for number of action, ranging from -1.72 to -8.13), consistent with shorter action sequences when user updates are incorporated. Token usage decreases for most models (e.g., Claude-Opus: **Avg.** ≈ -1061 tokens; DeepSeek: ≈ -330 tokens), although Haiku shows a marginal increase (**Avg.** $= +8.8$). Overall, these results suggest that informative interruptions can reduce action steps and often reduce token consumption, but the magnitude and even direction of token changes remain sensitive to both the model and the scenario.

F Evaluation on Different Interruption Position

In Table F.3, we evaluate model sensitivity to interruption position by injecting updates at $\{0, 0.2, 0.4, 0.6, 0.8\}$ of the no-update trajectory. This probes how the amount of committed intermediate state prior to interruption affects post-update adaptation. The position 0 setting serves as a full-information-at-start comparison, where the agent receives the update before taking any environment actions; comparing later positions against this setting helps separate the benefit of additional information from the cost of adapting to that information mid-execution. Overall, we observe a scale-dependent pattern. Large-scale models (e.g., Claude-Opus-4.5) achieve their best overall success when interruptions occur at later positions, indicating effective reuse of pre-interruption progress and localized trajectory repair. In contrast, smaller and open-weight models (e.g., Claude-Haiku, DeepSeek-V3.1) peak under early interruption and degrade as interruption is delayed, suggesting higher sensitivity to stale assumptions and weaker reconciliation of environment state with updated intent. Late interruptions require not only update integration but also correction of persistent UI state and previously executed actions. Larger models appear capable of repairing committed trajectories, whereas relatively smaller models exhibit trajectory entanglement under delayed updates.

G Prompt for Data Synthesis

To construct user query for evaluating interruptible agents, we designed four distinct data synthesis strategies that simulate realistic user behavior patterns during web navigation tasks. Each strategy targets a specific type of user interruption: adding missing details, revising incorrect information, retracting unnecessary constraints, or combining multiple update types in a multi-turn interaction. We employed Claude-Opus-4.5 to generate these synthetic variations from WebArena, ensuring that the transformed queries keep semantic equivalence with original intent while reflecting natural user communication patterns.

G.1 Addition Strategy

This prompt generates scenarios where users incrementally provide missing details to their initial incomplete query. The model transforms a complete task specification into a less detailed initial query and produces natural follow-up updates that add the missing information. This simulates common real-world scenarios where users start with a vague request and progressively clarify their requirements.

Scenario: Revision								
Model	Metric	S/F	F/S	S/S	F/F	No-int	With-int	Avg.
claude-haiku-4-5	# cases	2.00	55.00	6.00	102.00	165.00	165.00	–
	# action	30.00	8.27	4.00	22.06	19.08	16.90	−2.18
	# token	8348.17	2189.47	834.33	7257.06	5551.53	5347.54	−204.00
claude-sonnet-4-5	# cases	1.00	62.00	6.00	96.00	165.00	165.00	–
	# action	30.00	8.21	5.00	21.09	18.28	15.72	−2.56
	# token	12 628.67	1975.61	808.50	5352.85	4164.62	3962.67	−201.95
claude-opus-4-5	# cases	1.00	86.00	3.00	75.00	165.00	165.00	–
	# action	3.00	7.20	6.00	17.80	16.34	11.97	−4.37
	# token	639.00	1366.18	988.67	2732.62	2761.43	1976.02	−785.41
qwen3-235b-a22b	# cases	0.00	42.00	1.00	122.00	165.00	165.00	–
	# action	0.00	5.69	4.00	21.39	16.54	17.28	0.75
	# token	0.00	271.87	73.00	550.19	575.68	476.45	−99.24
deepseek.v3.1	# cases	1.00	41.00	2.00	121.00	165.00	165.00	–
	# action	30.00	8.56	3.00	17.21	16.00	14.96	−1.04
	# token	1455.33	786.34	47.00	1195.42	1130.22	1081.43	−48.79
mistral-large-3	# cases	0.00	25.00	4.00	136.00	165.00	165.00	–
	# action	0.00	6.64	11.00	14.71	12.22	13.39	1.17
	# token	0.00	3690.71	1337.08	3717.74	2509.71	3655.93	1146.22

Table D.1: **Efficiency breakdown under the revision scenario shown above.** We report action and token statistics grouped by paired outcomes between the interrupted run and the no-update run: S/F denotes interruption success and no-update failure; F/S denotes interruption failure and no-update success; S/S and F/F denote both succeed and both fail, respectively. **No-int** and **With-int** are averages over all episodes, and **Avg.** denotes the difference (With-int − No-int).

Scenario: Retraction								
Model	Metric	S/F	F/S	S/S	F/F	No-int	With-int	Avg.
claude-haiku-4-5	# cases	5.00	38.00	24.00	98.00	165.00	165.00	–
	# action	14.40	8.53	7.62	23.83	24.36	17.66	−6.70
	# token	7862.87	2976.55	3812.60	9956.60	7383.16	7391.96	8.80
claude-sonnet-4-5	# cases	8.00	29.00	35.00	93.00	165.00	165.00	–
	# action	19.38	8.72	6.51	24.16	22.88	17.47	−5.41
	# token	5444.88	2454.53	1793.08	8384.95	5938.83	5801.81	−137.02
claude-opus-4-5	# cases	6.00	41.00	44.00	74.00	165.00	165.00	–
	# action	6.83	6.98	7.07	18.34	20.22	12.09	−8.13
	# token	1098.78	1822.95	1755.70	3884.57	3764.03	2703.29	−1060.74
qwen3-235b-a22b	# cases	9.00	23.00	24.00	109.00	165.00	165.00	–
	# action	15.33	5.43	3.92	19.72	21.03	15.19	−5.84
	# token	343.00	137.78	101.62	604.45	607.57	452.00	−155.57
deepseek.v3.1	# cases	7.00	29.00	20.00	109.00	165.00	165.00	–
	# action	16.43	9.10	5.90	15.08	19.42	12.98	−6.45
	# token	938.62	918.14	548.75	1253.23	1425.77	1095.60	−330.18
mistral-large-3	# cases	10.00	21.00	11.00	123.00	165.00	165.00	–
	# action	9.60	8.38	15.09	13.54	14.47	12.75	−1.72
	# token	2249.03	2386.43	3136.42	3459.35	3514.49	3227.92	−286.58

Table E.2: **Efficiency breakdown under the retraction scenario shown above.** We report action and token statistics grouped by paired outcomes between the interrupted run and the no-update run: S/F denotes interruption success and no-update failure; F/S denotes interruption failure and no-update success; S/S and F/F denote both succeed and both fail, respectively. **No-int** and **With-int** are averages over all episodes, and **Avg.** denotes the difference (With-int − No-int).

G.2 Revision Strategy

This prompt creates scenarios where users initially provide incorrect information and subsequently correct their mistakes. The model generates plausible but incorrect details in the initial query (such as wrong dates, products, or quantities) and produces natural correction updates that fix these errors to recover the original intent. This reflects real-world situations where users realize and rectify their initial mistakes during interaction.

G.3 Retraction Strategy

This prompt generates scenarios where users initially over-specify their requirements with extra constraints or details, then remove these unnecessary elements. The model adds additional filters, requirements, or specifications to the original query

Model	Interrupt Position	Reddit	GitLab	CMS	Map	Shopping	Overall
Claude-Haiku-4.5	no-update	5.30	20.00	11.40	0.00	20.00	12.12
	0	21.10	50.00	45.70	11.50	40.00	35.15
	0.2	31.60	50.00	54.30	23.10	51.10	**43.64**
	0.4	26.30	36.70	37.10	15.40	33.30	29.70
	0.6	15.80	33.30	25.70	23.10	35.60	27.88
	0.8	21.10	36.70	17.10	15.40	31.10	24.24
Claude-Sonnet-4.5	no-update	0.00	20.00	2.90	7.70	28.90	13.33
	0	57.90	63.30	60.00	19.20	46.70	**47.88**
	0.2	47.40	66.70	60.00	19.20	46.70	47.27
	0.4	47.40	53.30	57.10	26.90	42.20	44.24
	0.6	47.40	53.30	57.10	19.20	35.60	41.21
	0.8	52.60	53.30	48.60	23.10	42.20	42.42
Qwen3-235B-A22B	no-update	5.30	16.70	2.90	0.00	11.10	7.27
	0	21.10	36.70	40.00	15.40	17.80	25.45
	0.2	26.30	43.30	45.70	11.50	28.90	31.52
	0.4	15.80	46.70	51.40	7.70	33.30	**32.73**
	0.6	15.80	36.70	40.00	7.70	33.30	27.27
	0.8	5.30	33.30	48.60	11.50	33.30	27.88
DeepSeek-V3.1	no-update	0.00	23.30	8.60	0.00	11.10	9.09
	0	31.60	36.70	28.60	3.80	28.90	26.06
	0.2	26.30	33.30	40.00	15.40	31.10	**30.30**
	0.4	26.30	36.70	37.10	15.40	33.30	29.70
	0.6	15.80	33.30	25.70	23.10	35.60	27.88
	0.8	21.10	36.70	17.10	15.40	31.10	24.24
Mistral-Large-3	no-update	0.00	13.30	8.60	0.00	11.10	7.27
	0	21.10	20.00	28.60	3.80	22.20	18.79
	0.2	10.50	26.70	25.70	11.50	33.30	22.42
	0.4	21.10	23.30	20.00	11.50	24.40	19.39
	0.6	10.50	30.00	20.00	11.50	31.10	21.21
	0.8	5.30	33.30	48.60	11.50	33.30	**27.88**
Claude-Opus-4.5	no-update	10.50	23.30	8.60	11.50	17.80	13.94
	0	63.20	70.00	65.70	23.10	57.80	54.55
	0.2	63.20	50.00	60.00	23.10	48.90	47.27
	0.4	63.20	60.00	68.60	34.60	60.00	55.76
	0.6	52.60	66.70	71.40	30.80	55.60	54.55
	0.8	57.90	63.30	68.60	38.50	62.20	**56.97**

Table F.3: **Task success rate under different interruption positions.** "no-update" denotes runs that do *not* receive the update. Overall, large-scale models improve under later interruptions, while relatively smaller models degrade as interruption is delayed, indicating higher sensitivity to stale/noisy intermediate state.

and produces natural retraction updates that remove these extras. This simulates situations where users start with overly restrictive criteria and progressively relax their constraints to broaden their search or simplify their request.

G.4 Multi-Turn Mixed Strategy

This prompt creates the most complex and realistic scenarios by combining all three update types (addition, revision, and retraction) within a single interaction sequence. The model generates an initial query that is simultaneously incomplete, partially incorrect, and over-constrained, then produces a series of mixed updates to recover the original intent. The updates are randomized in order and explicitly labeled by type, simulating natural multi-turn conversations where users iteratively refine their requests through various types of modifications.

Data Synthesis–Addition

```
Scenario: We are simulating an interruption scenario where a user may provide updates to
its initial query for an agent.

Task: Given the task_metadata, now revise the intent query by making {num_update} updates.

Requirement:
1. The transformed_initial_intent combined with the updates should have the same meaning
as the original intent query and result in the same ground truth answers.
2. Express the updates in a realistic user/customer tone and be concise.
3. Each update should be essential-lacking any one update should affect the ground truth
answers.


#####
Output:
[{
task_id: {task_id}
intent: {original_intent}
transformed_initial_intent: {transformed_initial_intent}
updates: [{update_1}, {update_2}, {update_3}]
}]


######
Input:
num_update: {num_of_update}
task_metadata: {task_metadata}
```

Data Synthesis–Revision

```
Scenario: We are simulating a modification interruption scenario where a user initially
provides an incorrect or mistaken query and then follows up with corrections/modifications
to fix the errors.

Task: Given the task_metadata, revise the intent query by:
Creating a transformed_initial_intent that changes {num_update} critical detail(s) from
the original intent (e.g., wrong date, wrong product, wrong quantity, wrong location, etc.)
Generating corresponding updates that correct those mistaken details to recover the
original intent.

Requirements:
1. The transformed_initial_intent should contain plausible but incorrect details compared
to the original intent.
2. The transformed_initial_intent combined with the updates should have the same meaning
as the original intent and result in the same ground truth answers.
3. Express the updates as realistic corrections in a natural user/customer tone.
4. Each correction should be essential-lacking any one update should result in different
ground truth answers than the original intent.


#####
Output:
[{task_i: {task_id}
intent: {original_intent}
transformed_initial_intent: {transformed_initial_intent}
updates: [{update_1}, {update_2}, {update_3}]
}]


######
Input:
num_update: {num_of_update}
task_metadata: {task_metadata}
```

Data Synthesis–Retraction

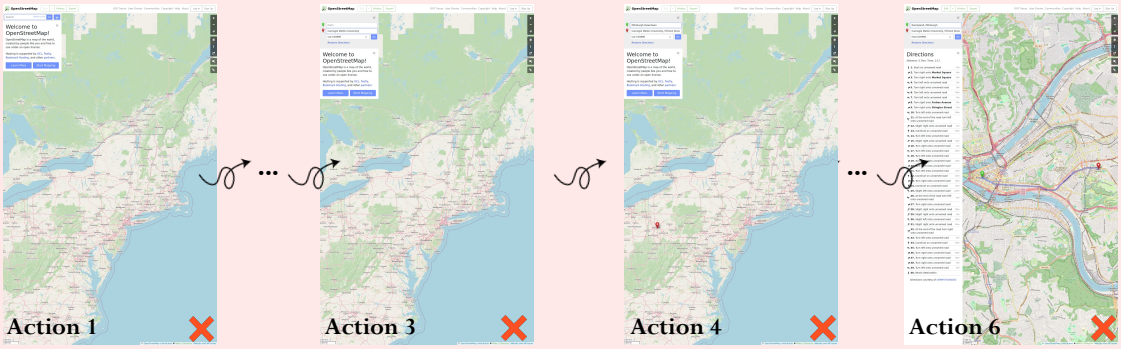
```
Scenario: We are simulating a retraction interruption scenario where a user initially
provides a query with extra details/constraints and then follows up to remove or retract
some of those details.

Task: Given the task_metadata, revise the intent query by:
Creating a transformed_initial_intent that adds {num_update} additional critical
detail(s)/constraint(s) to the original intent (e.g., extra filters, additional
requirements, more specifications, etc.)
Generating corresponding updates that remove/retract those extra details to recover the
original intent.

Requirements:
1. The transformed_initial_intent should contain additional plausible details/constraints
beyond the original intent.
2. The transformed_initial_intent combined with the updates should have the same meaning
as the original intent and result in the same ground truth answers.
3. Express the updates as realistic retractions in a natural user/customer tone.
4. Each retraction should be essential-lacking any one update should result in different
ground truth answers than the original intent.


#####
Output:
[{
task_id: {task_id}
intent: {original_intent}
transformed_initial_intent: {transformed_initial_intent}
updates: [{update_1}, {update_2}, {update_3}]
}]


######
Input:
num_update: {num_of_update}
task_metadata: {task_metadata}
```

Data Synthesis–Multi Turn

```
Scenario: We are simulating a mixed multi-turn interruption scenario where a user
initially provides a query and then follows up with a combination of:
Additions: Providing missing details that were not in the initial query
Modifications: Correcting mistaken/incorrect details from the initial query
Retractions: Removing extra details/constraints that were initially included


Task: Given the task_metadata, revise the intent query by:
Creating a transformed_initial_intent that differs from the original intent through a
combination of:
Missing critical detail(s) (to be added later)
Incorrect detail(s) (to be corrected later)
Extra detail(s)/constraint(s) (to be retracted later)
Generating {num_update} corresponding updates that recover the original intent through
additions, modifications, and/or retractions.

Requirements:
1. The transformed_initial_intent should be a plausible but
incomplete/incorrect/over-specified/mixed version of the original intent.
2. The transformed_initial_intent combined with all updates should have the same meaning
as the original intent and result in the same ground truth answers.
3. Express the updates in a realistic user/customer tone and be concise.
Each update should be essential-lacking any one update should affect the ground truth
answers.
4. The updates should include a mix of at three different types (addition, modification,
retraction). The order of types should be randomized.
Clearly label each update with its type in the updates_type array.


#####
Output:
[{
task_id: {task_id}
intent: {original_intent}
transformed_initial_intent: {transformed_initial_intent}
"updates_type": ["{type_1}", "{type_2}", "{type_3}"],
updates: [{update_1}, {update_2}, {update_3}]
}]


######
Input:
num_update: {num_of_update}
task_metadata: {task_metadata}
```

Intent: Compare the difference in time for walking and driving route to Carnegie Mellon University
True intent: Compare the difference in time for walking and driving route from Randyland to Carnegie Mellon University

Error Scenario



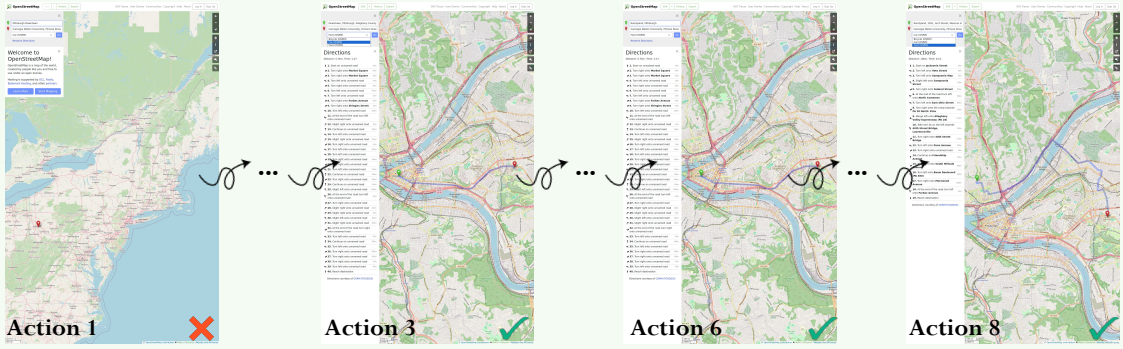
<think>I need walking vs driving times, so I should open the routing/directions feature first.<\think>
<answer>do(action="Click", element="15") **(Directions Icon)**
)</answer>

<think>Now I'll enter a starting point; I'll use "Pittsburgh Downtown".<\think>
<answer>do(action="Type", argument="Pittsburgh Downtown", element="13")</answer>

<think>Mode is Car; click Go for driving first, then switch to Foot for walking.<\think>
<answer>do(action="Click", element="16") **(Go Button)**
)</answer>

<think>Summarize: walking 77 min, driving 11 min, difference 66 min.<\think>
<answer>exit(message="...")</answer>

Correct Scenario



<think>I can see the directions form is set up with "Pittsburgh Downtown" in the "From" field and "Carnegie Mellon University" in the "To" field…</think>
<answer>do(action="Click", element="16") **(Go Button)**</answer>

<think>…Wait, the user mentioned they want to start from "Randyland" in their interrupt. I need to change the starting point from "Pittsburgh Downtown" to "Randyland"…</think>
<answer>do(action="Type", argument="Randyland", element="14")
</answer>

<think>…I see that I've typed "Randyland, Pittsburgh" in the From field, but the route hasn't been recalculated yet…</think>
<answer>do(action="Click", element="17")</answer>

<think>…Walking takes 91 minutes (or 1 hour 31 minutes) longer than driving. Let me provide the answer to the user.</think>
<answer>exit(message=…Walking: 7.9km, 1 hour 45 minutes (1:45)\n- Driving: 10km, 14 minutes (0:14)
…)</answer>

Figure G.1: User interruption with additional intent in WebArena. The agent initially plans a walking vs. driving comparison from Pittsburgh Downtown → Carnegie Mellon University, but a mid-trajectory user interrupt adds a new constraint (start from Randyland); we contrast an error trace that fails to fully retract/update the plan with a correct trace that revises the origin, recomputes routes, and returns the updated time difference.