The Long-Horizon Task Mirage?
Diagnosing Where and Why Agentic Systems Break

Anonymous ACL submission

Abstract

Despite rapid progress, LLM-based agents remain brittle on long-horizon tasks and their failure modes are still poorly understood, hindering principled diagnosis and cross-domain comparison. To address this gap, we introduce HORIZON, an initial cross-domain diagnostic benchmark for systematically constructing long-horizon tasks and analyzing the failure behaviors they elicit in LLM-based agents. Using HORIZON, we evaluate state-of-the-art (SOTA) agents from multiple model families (GPT-5 and Claude-4 variants), collecting 3100+ trajectories across four representative domains (web, database, operating system, and embodied) to study horizon-dependent degradation patterns. We further propose a trajectory-grounded LLM-as-a-Judge pipeline for scalable and reproducible failure attribution, and validate it against human annotation on trajectories with high agreement (inter-annotator κ=0.61; human–judge κ=0.84). Our findings indicate that performance does not gradually degrade but collapses sharply once horizons exceed a threshold, and that scaling base models alone is insufficient. Durable progress requires design-level advances in hierarchical planning, execution-time verification, and memory mechanisms. HORIZON provides a methodological foundation for systematic, cross-domain analysis of long-horizon agent failures and offers practical guidance for building more reliable agents.

1 Introduction

As large language models (LLMs) continue to advance (DeepSeek-AI et al., 2025; OpenAI, 2025a; Anthropic, 2025; Team et al., 2025a; Yang et al., 2025a), their use has expanded beyond conversational interfaces to agentic systems that interact with tools and environments in pursuit of real-world objectives (Team et al., 2025b; Xi et al., 2025a). When executing a complex task, an LLM-based agent must interpret the current state, select appropriate actions, and continually update its internal beliefs and plans in response to environmental feedback (Zhou et al., 2024b). However, in long-horizon settings where success depends on extended and interdependent sequences of actions, agents often fail in systematic yet non-obvious ways (Table 3), with performance degrading sharply as the task horizon increases (Yang et al., 2025b; Chen et al., 2025a; Xi et al., 2025b). For example, an embodied agent may succeed reliably on single-step manipulation (e.g., moving a colored cube), but fail completely once the task requires three sequential steps. This degradation is not merely additive: even a small per-step error rate compounds across dependent steps, driving agents from reliable short-task performance to near-systematic failure at longer horizons.

Understanding these failures is of practical and scientific importance, as many high-impact deployments of LLM-based agents are inherently long-horizon. Tasks such as debugging multi-file software systems, executing operating-system workflows, conducting iterative data analysis, and performing embodied manipulation require maintaining goals, constraints, and intermediate states over dozens or even hundreds of effective actions. Despite this prevalence, research on long-horizon agents remains fragmented: benchmarks are domain-specific, horizon definitions are inconsistent, and evaluations emphasize aggregate success rates, offering limited insight into how and why failures arise as horizons grow (Liu et al., 2023c; Trivedi et al., 2024; Kapoor et al., 2025; Zhang et al., 2025b; Chen et al., 2024). Consequently, findings are difficult to compare across domains and studies, and the field lacks a shared foundation for diagnosing long-horizon breakdowns.

A central challenge in studying long-horizon agent failures is that long-horizon behavior is inherently domain-dependent. Different domains impose distinct horizon scales and give rise to qual-

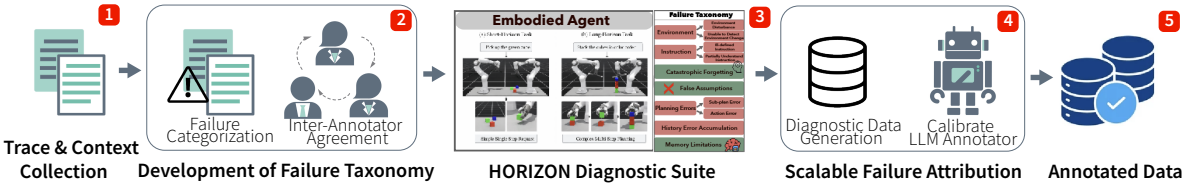

Figure 1: *The* HORIZON *diagnostic pipeline for scalable long-horizon failure analysis.* The pipeline consists of trace and context collection, taxonomy development with inter-annotator agreement, construction of the HORIZON diagnostic suite (unified horizon measurement and failure attribution), calibration of an LLM-based annotator, and large-scale failure annotation. This figure highlights our systematic approach to approximating the breaking-point problem in long-horizon agents, enabling scalable and reproducible analysis for future cross-domain research.

itatively different failure mechanisms, rendering any single, universal definition insufficient. For example, embodied agents may exhibit abrupt performance breakdowns on tasks requiring only a small number of sequential actions, whereas web-based agents can remain robust at comparable step counts. Conversely, web tasks often involve longer dependency chains spanning multiple pages, forms, and verification steps. Such cross-domain discrepancies fundamentally complicate efforts to characterize a unified breaking point for long-horizon agents. We study two core research questions for long-horizon agent reliability: (1) *RQ1* Where do agents break down as task horizons increase? and (2) *RQ2* Why do these failures emerge?

Addressing these questions requires a principled diagnostic framework that explains *where* and *why* agentic systems fail as horizons grow. Figure 1 illustrates the overall HORIZON diagnostic pipeline. We introduce HORIZON (**H**olistic **O**bservations for **R**easoning and fa**I**lure analy**Z**is in l**O**ng-horizo**N** agents), an initial cross-domain diagnostic benchmark that constructs task families with systematically increasing step requirements and enables structured analysis of horizon-dependent degradation, guided by agent workflow (Figure 2) and informed by Failure Mode and Effects Analysis (FMEA) (AIAG and VDA, 2019). We then evaluate state-of-the-art agent models (GPT-5 and Claude-4 model families) on 3100+ trajectories across four representative domains (Web, OS, Embodied, and Database), providing a cross-domain view of where performance breaks down. Finally, we analyze failed trajectories with a trajectory-grounded LLM-as-a-Judge pipeline to identify why failures emerge and how failure composition shifts with horizon. Since long-horizon rollouts yield very long raw traces, we validate the judge against expert labels on a pilot set of 40 trajectories. On this set, inter-annotator agreement is κ=0.61 and human–judge agreement is κ=0.84.

Our analysis shows two complementary findings. First, performance does not degrade gradually as task horizons grow; instead, agents exhibit a sharp drop in success once horizons exceed a model-specific threshold, indicating a discrete failure regime rather than a smooth decline. Second, the composition of failures shifts qualitatively across the horizon, where planning-related failures (Planning Error) and memory-related failures (Memory Limitation and Catastrophic Forgetting) come to dominate the error distribution in the longer horizon. Together, these findings suggest that scaling base models alone is insufficient for reliable long-horizon behavior, and durable progress instead requires design-level advances in hierarchical planning, execution-time verification, and memory mechanisms.

The contributions of this paper are as follows:

- We introduce HORIZON, an initial cross-domain diagnostic benchmark for systematically constructing long-horizon task families and analyzing horizon-dependent agent failure.
- We conduct a large-scale empirical study of 3100+ trajectories spanning four agentic domains (web, database, operating system, embodied) and multiple state-of-the-art model families (GPT-5 and Claude-4 model families), revealing consistent cross-domain degradation patterns.
- We develop a trajectory-grounded LLM-as-a-Judge pipeline for scalable, reproducible failure attribution, and validate it against human annotation (inter-annotator κ=0.61; human–judge κ=0.84).
- We identify two findings with direct design implications for future agentic AI systems,

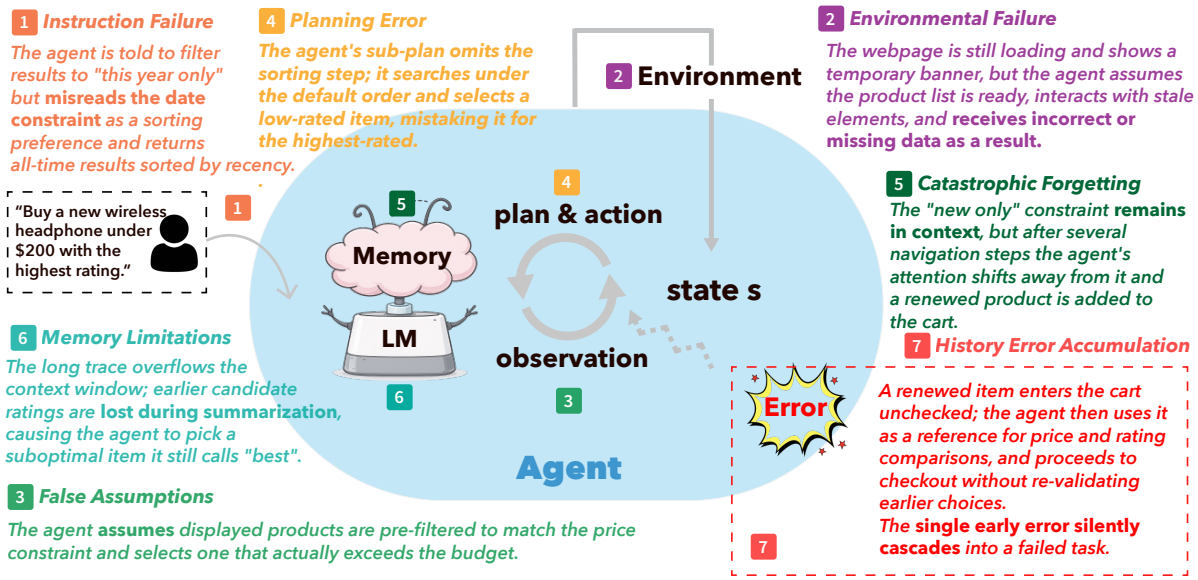

Figure 2: *Illustration of general agent execution and failure propagation.* Given an instruction 1, the agent iterates a standard loop: it observes the environment 2 to obtain observations 3, plans and selects an action 4 drawing on memory-backed context 6, executes that action to change the environment, and then updates its internal state 5 through memory 6. Failures can originate at any stage and compound across cycles 7. The seven legend boxes map directly to the failure categories in Table 1.

motivating method-design improvements beyond base-model scaling.

2 Related Work

Most benchmarks report success rates rather than treating horizon as a controlled variable or providing trajectory-level failure attribution. We briefly cover recent work on benchmarks and failure analysis for agentic AI systems. Further related work see Appendix A.

Interactive benchmarks and agent evaluation. Standard suites span web navigation (WebArena (Zhou et al., 2024a), Mind2Web (Deng et al., 2023)), multimodal web tasks (Visual WebArena (Koh et al., 2024)), software engineering (SWE-bench (Jimenez et al., 2023)), app-style assistants (AppWorld (Trivedi et al., 2024)), databases (BIRD (Li et al., 2023), Spider (Yu et al., 2018)), and embodied instruction following (ALFWorld (Shridhar et al., 2021)). AgentBench (Liu et al., 2023c) and AgentGym (Xi et al., 2024) aggregate many environments under one evaluation protocol. Newer work stressing harder or longer interaction (e.g., BrowseComp (Wei et al., 2025), MIRAGE-Bench (Zhang et al., 2025b), holistic leaderboards (Kapoor et al., 2025), AgentGym-RL (Xi et al., 2025b)) still foregrounds terminal success, while two concurrent benchmarks (Toolathlon (Luo et al., 2025) and UltraHorizon (Luo et al., 2025)) cover only limited domains. We instead vary intrinsic horizon H^* and analyze how the mix of failure types shifts as horizon grows across four heterogeneous, real-world-grounded domains.

Failure analysis and diagnostics. Existing failure studies are typically either domain-specific (covering code (Chen et al., 2023), web (Koh et al., 2024), and robotics (Zeng et al., 2020)) or remain coarse at the benchmark level (Liu et al., 2023c). While some recent works have examined multi-agent failure modes (Zhang et al., 2025a; Yu et al., 2025; Cemri et al., 2025) or short-horizon settings (Zhu et al., 2025), we target long-horizon failures in both single-agent and multi-agent systems in this paper.

3 HORIZON

In this section, we introduce HORIZON, an initial cross-domain diagnostic benchmark for systematic analysis of long-horizon agent behavior. HORIZON combines agent-independent task-structure metrics with controlled task extension and structured failure attribution, enabling principled analysis of when and why agents fail as task horizons increase.

Risk Level	Category	Definition
Process-level (PFMEA)	Environment Error	The external world changes, degrades, or behaves stochastically in ways not modeled by the agent's plan, *or* the agent cannot reliably perceive such a change, so its internal belief diverges from the actual state.
	Instruction Error	The task specification is ambiguous, under-specified, or misaligned with environment capabilities, *or* the agent grasps the high-level intent but fails to internalize critical constraints, modifiers, or exceptions.
	Planning Error	The agent mis-decomposes the goal into incorrect, incomplete, or poorly ordered sub-steps (sub-plan), *or* selects/executes an incorrect concrete action despite an appropriate high-level plan (action).
	History Error Accumulation	No single step's error is sufficient on its own, but minor errors propagate and compound across steps, causing trajectory-level failure.
Design-level (DFMEA)	Memory Limitation	Task-relevant information exceeds the agent's effective context capacity and is dropped from context, causing failure.
	Catastrophic Forgetting	The agent fails to apply previously acquired knowledge, constraints, or decisions that remain in context.
	False Assumption	The agent commits to an incorrect premise about the world or task state and proceeds on a false basis without verification.

Table 1: *Formal definitions of the seven failure categories in our taxonomy.* Categories are organized by the two FMEA-inspired risk levels. Per-domain instantiations and representative examples are in Appendix C.

3.1 Defining Task Horizons

A fundamental challenge in studying long-horizon agent failures is separating task complexity from agent inefficiency. For example, an agent executing 50 actions on a simple task while repeatedly failing on a sub-task is not solving a long-horizon problem, but repeatedly failing on a short-horizon task. To enable principled cross-domain analysis while respecting domain-specific characteristics, we propose a two-layer approach: *theoretical metrics* that capture task structure independent of any agent, and *technical implementations* that operationalize these metrics through systematic task extension.

Theoretical Definition: Agent-Independent Task Characterization. We define two complementary metrics that characterize the intrinsic long-horizon nature of tasks:

Intrinsic Horizon (H^*) measures the minimum number of effective actions required by an optimal policy to complete the task. This can be established through expert demonstrations, oracle solvers, or formal task specifications. For example, the web task "find and purchase wireless headphones under $200" has $H^* = 8$ regardless of how many attempts any particular agent requires: $H^* = |\{a_1 : \text{search}, a_2 : \text{filter price}, a_3 : \text{sort rating}, \ldots, a_8 : \text{checkout}\}|$.

Compositional Depth (s) measures the number of high-level subtasks that make up the task, including sequential sub-goals, conditional branches, and parallel goals to be jointly maintained. s reflects planning complexity beyond linear execution: $s = \max_{p \in \text{paths}} |\text{decision_nodes}(p)|$. A task requiring "if condition A holds, execute plan B, otherwise execute plan C, then merge results" has $s = 3$, while a purely sequential workflow has $s = 1$.

Technical Implementation: Controlled Horizon Extension. To empirically study how agent performance degrades with increasing horizon, we systematically vary H^* by increasing the compositional depth s defined above through controlled task construction. We design two extension methods to ensure $H^*(s)$ is monotonically increasing and positively correlated with s, though the exact mapping depends on domain characteristics.

Depth Extension incrementally adds intermediate steps between existing actions. Each extension operation ($s \rightarrow s + 1$) inserts one or more necessary sub-tasks that an optimal policy cannot skip. For a baseline task with $s_{\text{base}} = 1$, depth extension to level s produces: $H^*_{\text{depth}}(s) = H^*_{\text{base}} + \Delta(s)$, where $\Delta(s)$ is monotonically increasing in s. The growth function $\Delta(s)$ depends on domain structure: in OS, each extension typically adds 1-2 permission checks; in Database, each extension may add 2-3 filtering operations. The key property is $\Delta(s_2) > \Delta(s_1)$ whenever $s_2 > s_1$, ensuring systematic horizon growth. This approach is particularly effective for domains where the initial environment state is fixed (OS, Database).

Breadth Extension combines k independent baseline tasks into a single composite workflow. For baseline tasks $\{t_1, \ldots, t_k\}$ each at $s_i = 1$ with

intrinsic horizons $\{H_1^*, \ldots, H_k^*\}$, the combined task has $s = k$ subtasks and: $H^*_{\text{breadth}}(s) = \sum_{i=1}^{s} H_i^* + \epsilon(s)$, where $\epsilon(s) \geq 0$. The coordination overhead $\epsilon(s)$ accounts for additional actions required to manage multiple goals, including task switching, inter-task verification, and state tracking across sub-tasks. For instance, combining two embodied manipulation tasks may require intermediate verification steps ("check if yellow cylinder was accidentally displaced while stacking cubes") and context re-establishment ("verify red cube is still properly stacked before proceeding to sphere alignment"). This approach better captures real-world scenarios requiring parallel goal maintenance and is more suitable for domains with variable initial states (Web, Embodied).

3.2 Failure Attribution: 7-Category Taxonomy

We systematically diagnose why agents fail through structured failure attribution and derive a seven-category taxonomy that captures recurring failure modes across long-horizon tasks, providing actionable diagnostics for agent improvement.

Taxonomy Overview. We acknowledge that long-horizon failures can partly stem from fundamental limitations of base LLMs, such as hallucination or weak instruction following. However, in developing HORIZON, we focus on failure patterns that emerge from long-horizon dynamics that can be addressed by improvements in agent design, independently of or complementary to advances in the base models themselves. Table 1 lists the seven resulting failure categories with their formal definitions. At the trajectory level, the seven categories represent orthogonal dimensions of agent behavior rather than mutually exclusive classes, and a single failed trajectory may exhibit several categories across different steps. Representative examples across all four domains and complete per-domain definitions are provided in Appendix C. The decision procedure for attributing a given failure step to one of these categories is illustrated in appendix Figure 8.

Why It Works? Our taxonomy is grounded in three validation pillars that demonstrate its reliability as a diagnostic framework:

Construction: Theory and Literature Grounding. We construct the seven-category taxonomy from a hierarchical risk-modeling perspective inspired by **Failure Mode and Effects Analysis** (FMEA), and ground it in a **structured literature review** spanning web, OS, embodied, database, and related agentic settings. We conceptualize long-horizon failure as two interacting levels of risk, using an agent-oriented reading of PFMEA and DFMEA. *Process-level risks (PFMEA)* arise during sequential rollout: environment interaction, execution-time failures to follow or interpret the task specification (instruction), flawed subplanning or action ordering (planning error), and compounding mistakes carried across prior steps (history error accumulation). *Design-level risks (DFMEA)* reflect limitations of the engineered agent: memory and context mechanisms (memory limitation, catastrophic forgetting of constraints) and persistent mismatch between internal beliefs and the task or observations (false assumptions). We then extract failure descriptions from prior work and iteratively refine categories through open coding, consolidation, and boundary refinement. The final categories are retained based on cross-domain coverage, interpretability, and annotation consistency; full paper-to-category mappings and coding examples are provided in Appendix B.

Empirical Grounding in Agent Workflow. The taxonomy is empirically grounded in the agent execution loop, where planning, action, environment interaction, observation, and state update jointly define trajectory evolution. Figure 2 illustrates how distinct failure types arise at different stages of this loop and how early errors can propagate and compound over time, providing a mechanism-level basis for structured failure attribution.

Experimental Validation. We validate the practical applicability of HORIZON through both human annotation and trajectory-grounded LLM-as-a-Judge analysis of failed rollouts (Section 4). In a pilot study, two expert annotators independently labeled a shared set of 40 failure trajectories, reaching substantial inter-annotator agreement ($\kappa = 0.61$) and confirming that the taxonomy categories are consistently interpretable. We further validated the LLM-as-a-Judge pipeline against one of the human annotators on the same 40 traces, obtaining strong agreement ($\kappa = 0.84$). We acknowledge that 40 trajectories is a modest sample, and that human–judge agreement exceeds inter-human agreement. Both arise from a finite-resource trade-off and from the methodological framing of the judge—rather than from sample bias. A full discussion of sample-size choice, an agreement-vs. sample-size stability analysis, and the H–H vs. H–J gap is given in Appendix F.

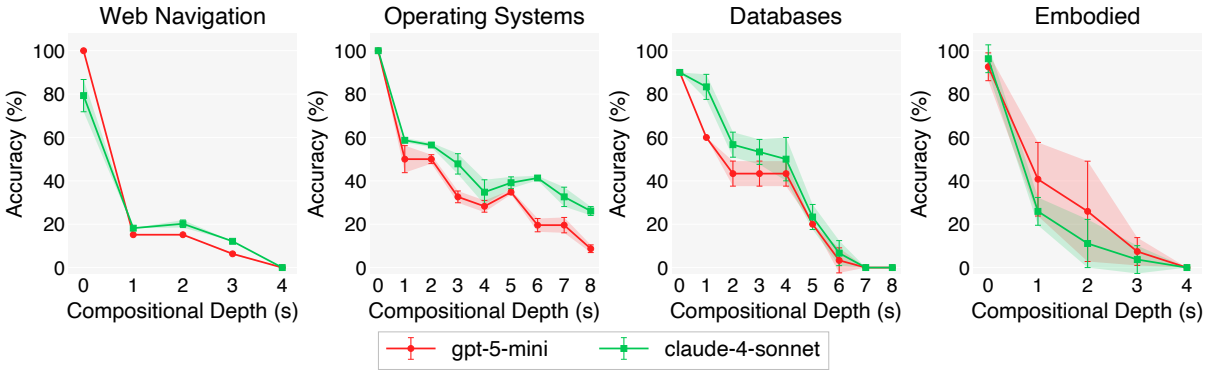

Figure 3: *Current model performance as a function of horizon extension.* Plots show success rate (accuracy) versus the compositional depth s, where s denotes the HORIZON-defined extension level corresponding to the number of high-level subtasks composed within a task. Note that s measures compositional depth, not raw action count. Each unit of s expands to multiple low-level actions during execution, with per-trajectory action budgets of max 150 (Web), 50 (OS), 31 (DB), and 34 (Embodied) steps. Accuracy is computed over task sets with the same s. Each point reports the mean over three independent runs with identical task sets; variability across runs is shown as $\pm$ one standard deviation (error bars). We evaluate GPT-5-mini and Claude-4-Sonnet across all domains. Across models and domains, performance consistently decreases and exhibits a sharp drop beyond a certain value of s, empirically supporting the validity of our horizontal horizon definition. More details are provided in Section 4.

4 Empirical Performance of SOTA Models

We characterize each agent's long-horizon capacity by its breaking region: the range of horizons over which task success transitions sharply from high to low, and failures shift from recoverable local errors to irreversible trajectory-level derailment. This region is not a universal constant. Different agents have markedly different long-horizon capacities, and even the same agent can break at very different horizons across domains, depending on factors such as environment dynamics, observability, and action semantics.

Experimental setup. Using HORIZON, we evaluate agents powered by SOTA foundation models (GPT-5 and Claude-4 model families) on more than 700 tasks spanning four established sources: WebArena (Zhou et al., 2024a), AgentBench (Liu et al., 2023c), MAC-SQL (Wang et al., 2025a), and Isaac Sim (Kachaev et al., 2025), yielding 3100+ trajectories in total. Our objective is to characterize the qualitative trend of horizon-induced degradation in SOTA agents. Under a fixed compute budget, we first adopt GPT-5-mini as the common reference across all four domains for cross-domain consistency, with an additional cross-model check confirming the same degradation pattern (**Experiment 1**). Then, to validate the reliability of the evaluation pipeline, we benchmark our trajectory-grounded LLM-as-a-Judge (GPT-5) against human annotations on failure trajectories (**Experiment 2**).

To reduce confounding, we construct nested task sets across horizon levels: the set at level (s+1) contains all tasks from level s plus additional tasks requiring longer execution. Thus, comparisons across adjacent levels are controlled by construction, and the primary varying factor is horizon length (and its induced long-range dependencies), rather than a change in task family or evaluation protocol. To further isolate the horizon effect from base-task difficulty, we restrict the task pool to benchmark tasks on which the GPT-5-mini baseline achieves roughly 90% success, and then horizon-extend these base tasks via HORIZON to construct deeper levels. This filtering choice does not drive the observed trend, nor does it bias the cross-model comparison: the same monotonic degradation holds on unfiltered task sets. Additional details of Experiment 1 are provided in Appendix D, its results in Appendix E, and the details and results of Experiment 2 in Appendix F.

Main Results. Horizontally, as shown in Figure 3, agent performance exhibits three consistent cross-domain patterns as the horizon extension level s increases. First, performance degrades non-linearly with increasing s. Across all domains, success rates remain relatively stable or decline gradually at small s, rather than decreasing proportionally with each additional subtask. Second, all domains exhibit a sharp performance drop beyond small s, where success transitions abruptly from

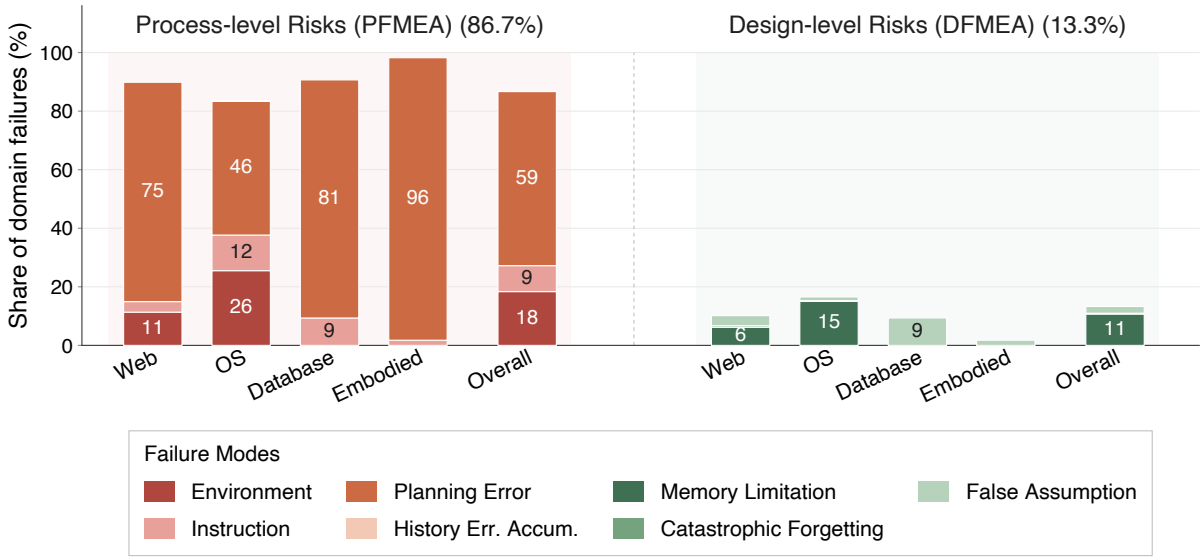

Figure 4: *Distribution of failure modes across four task domains on 3100+ trajectories.* Bars show the proportion of each failure type relative to the total failed traces within each domain. Failures are grouped into Process-level Risks (PFMEA) (environment, instruction, planning errors, and history error accumulation) and Design-level Risks (DFMEA) (memory limitation, catastrophic forgetting, and false assumptions). More details are provided in Section 4.

partial robustness to near-systematic failure, instead of degrading smoothly across the entire range of s. Third, the same extension level s induces markedly different task difficulty across domains: Web collapses at very small s, OS and Database domains sustain moderate performance until later extension levels, and Embodied tasks degrade steeply even with minimal increases in s. This variation highlights domain-specific breaking behavior in both the location and severity of performance collapse. Finally, in the Web, OS, and Database domains, performance gaps between models narrow substantially after entering the breaking region, as success rates converge toward low values, indicating diminished model differentiation once long-horizon failure dominates.

Vertically, as shown in Figure 4, failed trajectories across all domains can be systematically attributed to our seven-category failure taxonomy. In our experiments, both LLM-as-a-Judge and human annotators consistently apply these categories to long-horizon failures across all four domains, yielding reliable and interpretable annotations, indicating that our taxonomy provides comprehensive coverage of observed trajectory failures. Moreover, the result also show that failures are far from uniformly distributed: three categories consistently dominate at long horizons, namely *Planning Error*, *Catastrophic Forgetting*, and *Memory Limitation*, while the remaining categories account for a smaller share. Additional trajectories and LLM-as-a-Judge results are provided in Appendix E and Appendix F.

5 Actionable Steps Toward Responsible Long-Horizon Agent Development

Building on these results, we outline three actionable steps below that translate our diagnostic results into practices the community can adopt.

Unified Horizon Task Construction and Measurement Across Domains. Current agent evaluations adopt heterogeneous task designs and horizon definitions, often reporting success rates under incompatible notions of task length or complexity. This fragmentation makes cross-domain comparison difficult and obscures where long-horizon failures truly arise. We suggest that future work consider unified horizon-aware task construction and standardized horizon metrics, as piloted in HORIZON, when building and evaluating long-horizon benchmarks across domains. Two concrete advantages follow from this unification, both of which we demonstrate in our experiments. First, it enables principled cross-domain comparison: when domain A requires intrinsic horizon $H^* = 15$ to reach 60% success while domain B requires only $H^* = 8$, the comparison directly quantifies an agent's relative long-horizon capacity

across domains, which incompatible task definitions cannot support. Second, it enables targeted diagnosis: controlled horizon growth lets researchers localize the regime in which a particular model breaks down, rather than reporting only aggregate success rates. Together, these properties support reproducibility and cumulative verification and comparison across benchmarks and model families.

Scalable Failure Attribution via LLM-as-a-Judge. Beyond measuring when long-horizon agents fail, understanding why they fail is critical for targeted improvements. However, given the length and volume of long-horizon agent trajectories, exhaustive human attribution is impractical. Recent work has successfully applied LLM-as-a-Judge to related agent-evaluation settings (Zheng et al., 2023; Kapoor et al., 2025). Building on this paradigm, we propose systematic failure attribution using LLM-based judges grounded in our seven-category taxonomy (Table 1). LLM-as-a-Judge for failure attribution offers scalability (annotating thousands of trajectories is infeasible for humans), consistency (it reduces annotator variance on fine-grained categories), and actionability (structured taxonomy-aligned outputs map directly to mitigation strategies). We suggest that future work adopt an LLM-as-a-Judge pipeline to enable large-scale diagnosis, with a human-validated subset calibrating reliability.

Implications for Future Agentic AI Systems. Although some long-horizon failures can be attributed to imperfect base-model capabilities, our results suggest that the dominant failure mechanisms in long-horizon agents are unlikely to be resolved by model scaling alone. As shown in Figure 4, planning-related failures, memory limitations, and catastrophic forgetting account for a substantial share of failed trajectories across domains. Planning failures are particularly damaging because they arise early, propagate through downstream actions, and turn recoverable local mistakes into irreversible trajectory-level derailments. As horizons grow, sub-planning decisions expand combinatorially, making globally coherent plan selection harder and early errors increasingly path-dependent. Because such errors are costly to roll back once committed, addressing them requires not only better upfront planning but also runtime mechanisms that can detect and repair plan deviations as they occur. Memory limitations and catastrophic forgetting become more pronounced for related reasons. As trajectory length and context load grow, finite memory budgets and attention constraints force agents into a trade-off between retaining long-range constraints and incorporating new observations, increasing the risk of losing critical earlier information. Together, these patterns point to three concrete design priorities for future agentic systems: *(i)* hierarchical sub-planning to contain the combinatorial blow-up of long-horizon decisions; *(ii)* execution-time plan verification and repair to catch early errors before they propagate irreversibly; and *(iii)* memory mechanisms that preserve and re-surface long-range constraints under bounded context. These are architectural directions rather than scale-driven ones, and we view them as the more promising path toward reliable long-horizon agents.

6 Conclusion

Long-horizon reliability is increasingly the bottleneck for deploying LLM agents in real-world settings, yet the community lacks shared tools for diagnosing where and why agents break down as tasks lengthen. Our work contributes toward closing this gap through three artifacts: (1) HORIZON, a cross-domain benchmark built on a unified horizon definition and systematic extension procedures across four domains; (2) a seven-category failure taxonomy; and (3) a trajectory-grounded LLM-as-a-Judge attribution pipeline validated against human annotations.

Across 3100+ trajectories spanning multiple model families, our analysis shows that long-horizon breakdown is not merely a drop in task success, but a structural shift in failure composition as horizon grows. Planning-related errors and memory-related errors emerge as dominant bottlenecks across domains, suggesting that improving base-model capability alone is unlikely to fully address these failures. These findings point toward a design-centered research agenda for future agentic systems, including hierarchical and constraint-aware planning, execution-time plan verification and repair, and stronger long-range memory mechanisms. We envision HORIZON as a starting point for systematically diagnosing long-horizon failures, and hope it equips the community with a foundation for building principled, trustworthy agentic systems.

Limitations

We view HORIZON as an initial diagnostic benchmark with limitations that point to natural extensions. Our seven-category taxonomy characterizes what goes wrong at each failed step but does not separately capture complementary behavioral dimensions such as recovery and self-correction, action efficiency, failure compositionality, and grounding/verification behavior. We treat these as orthogonal axes along which HORIZON can be extended in future work.

Ethical Considerations

We do not foresee ethical concerns arising from this work: all tasks in HORIZON are synthetic or drawn from publicly released agent benchmarks and contain no personally identifying information, and all failure annotations were performed by the authors.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, and 1 others. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

AIAG and VDA. 2019. *AIAG & VDA FMEA Handbook*. Automotive Industry Action Group.

Anthropic. 2025. Claude-4-sonnet.

Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. 2025. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2838–2845. IEEE.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. Why do multi-agent llm systems fail? *Preprint*, arXiv:2503.13657.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. 2025a. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

Yanan Chen, Ali Pesaranghader, Tanmana Sadhu, and Dong Hoon Yi. 2024. Can we rely on llm agents to draft long-horizon plans? let's take travelplanner as an example. *arXiv preprint arXiv:2408.06318*.

Zixuan Chen, Jing Huo, Yangtao Chen, and Yang Gao. 2025b. Robohorizon: An llm-assisted multi-view world model for long-horizon robotic manipulation. *arXiv preprint arXiv:2501.06605*.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*.

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*.

Tongtong Feng, Xin Wang, Zekai Zhou, Ren Wang, Yuwei Zhan, Guangyao Li, Qing Li, and Wenwu Zhu. Evoagent: Self-evolving agent with contin-ual world model for long-horizon tasks.

Gonzalo Gonzalez-Pumariega, Leong Su Yean, Neha Sunkara, and Sanjiban Choudhury. 2025. Robotouille: An asynchronous planning benchmark for llm agents. *arXiv preprint arXiv:2502.05227*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Nikita Kachaev, Andrei Spiridonov, Andrey Gorodetsky, Kirill Muravyev, Nikita Oskolkov, Aditya Narendra, Vlad Shakhuro, Dmitry Makarov, Aleksandr I Panov, Polina Fedotova, and 1 others. 2025. Mind and motion aligned: a joint evaluation isaacsim benchmark for task planning and low-level policies in mobile manipulation. *arXiv preprint arXiv:2508.15663*.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. 2025. Acon: Optimizing context compression for long-horizon llm agents. *arXiv preprint arXiv:2510.00615*.

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, and 1 others. 2025. Holistic agent leaderboard: The missing infrastructure for ai agent evaluation. *arXiv preprint arXiv:2510.11977*.

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. *arXiv preprint arXiv:2401.13649*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2023. Bird: A big bench for large-scale database grounded text-to-sql evaluation. *arXiv preprint arXiv:2305.03111*.

Zhuoling Li, Xiaogang Xu, Zhenhua Xu, SerNam Lim, and Hengshuang Zhao. 2024. Larm: Large autoregressive model for long-horizon embodied intelligence. *arXiv preprint arXiv:2405.17424*.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023a. Llm+p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023b. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

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

Haotian Luo, Huaisong Zhang, Xuelin Zhang, Haoyu Wang, Zeyu Qin, Wenjie Lu, Guozheng Ma, Haiying He, Yingsha Xie, Qiyang Zhou, Zixuan Hu, Hongze Mi, Yibo Wang, Naiqiang Tan, Hong Chen, Yi R. Fung, Chun Yuan, and Li Shen. 2025. Ultrahorizon: Benchmarking agent capabilities in ultra long-horizon scenarios. *Preprint*, arXiv:2509.21766.

Aman Madaan, Amir Yazdanbakhsh, Jaemin Zhou, Samir Mishra, Jian Gupta, Yejin Song, Karthik Narasimhan, and 1 others. 2022. Memory-assisted prompt editing to improve gpt-3 after deployment. *arXiv preprint arXiv:2201.06009*.

OpenAI. 2025a. Gpt-5.

OpenAI. 2025b. Gpt-5-mini.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*.

Ofir Press, Noah A Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Shuzheng Si, Haozhe Zhao, Kangyang Luo, Gang Chen, Fanchao Qi, Minjia Zhang, Baobao Chang, and Maosong Sun. 2025. A goal without a plan is just a wish: Efficient and effective global planner training for long-horizon agent tasks. *arXiv preprint arXiv:2510.05608*.

Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. 2025. The illusion of diminishing returns: Measuring long horizon execution in llms. *arXiv preprint arXiv:2509.09677*.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. 2023. Adaplanner: Adaptive planning from feedback with language models. *arXiv preprint arXiv:2305.16653*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and et al. 2025a. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025b. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. *arXiv preprint arXiv:2407.18901*.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2025a. Mac-sql: A multi-agent collaborative framework for text-to-sql. *Preprint*, arXiv:2312.11242.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Weixuan Wang, Dongge Han, Daniel Madrigal Diaz, Jin Xu, Victor Rühle, and Saravan Rajmohan. 2025b. Odysseybench: Evaluating llm agents on long-horizon complex office application workflows. *arXiv preprint arXiv:2508.09124*.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025c. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2024. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025a. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, and 1 others. 2025b. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *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. *arXiv preprint arXiv:2410.13825*.

Yi Yang, Jiaxuan Sun, Siqi Kou, Yihan Wang, and Zhijie Deng. 2025b. Lohovla: A unified vision-language-action model for long-horizon embodied tasks. *arXiv preprint arXiv:2506.00411*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and 1 others. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Yifan Yu, Moyan Li, Shaoyuan Xu, Jinmiao Fu, Xinhai Hou, Fan Lai, and Bryan Wang. 2025. Correct: Condensed error recognition via knowledge transfer in multi-agent systems. *Preprint*, arXiv:2509.24088.

Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. 2020. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. 2025a. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. *Preprint*, arXiv:2505.00212.

Weichen Zhang, Yiyou Sun, Pohao Huang, Jiayue Pu, Heyue Lin, and Dawn Song. 2025b. Mirage-bench: Llm agent is hallucinating and where to find them. *arXiv preprint arXiv:2507.21017*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others.

2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and 1 others. 2024a. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024b. Archer: Training language model agents via hierarchical multi-turn rl. *Preprint*, arXiv:2402.19446.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. 2025. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, Xiaoteng Ma, Xiaodong Yu, Gowtham Ramesh, Jialian Wu, Zicheng Liu, Pan Lu, James Zou, and Jiaxuan You. 2025. Where llm agents fail and how they can learn from failures. *Preprint*, arXiv:2509.25370.

A Appendix: Extended Related Work

Most agent benchmarks report success rates, but fewer treat horizon as a controlled independent variable or provide structured, trajectory-level failure attribution that can be compared across domains. We organize related work into interactive benchmarks, training and planning methods, memory and long-context effects, and failure analysis.

A.1 Interactive benchmarks and agent evaluation

Standard suites span web navigation (WebArena (Zhou et al., 2024a), Mind2Web (Deng et al., 2023)), multimodal web tasks (Visual WebArena (Koh et al., 2024)), software engineering (SWE-bench (Jimenez et al., 2023)), app-style assistants (AppWorld (Trivedi et al., 2024)), databases (BIRD (Li et al., 2023), Spider (Yu et al., 2018)), and embodied instruction following (ALFWorld (Shridhar et al., 2021)). AgentBench (Liu et al., 2023c) and AgentGym (Xi et al., 2024) aggregate many environments under one evaluation protocol.

Newer benchmarks stress harder browsing, longer interaction, or richer diagnostics, including BrowseComp (Wei et al., 2025), MIRAGE-Bench (Zhang et al., 2025b), holistic leaderboards (Kapoor et al., 2025), and multi-turn RL with AgentGym-RL (Xi et al., 2025b). The emphasis in most of this work is still whether the task is solved at the end. We instead sweep intrinsic horizon H^* and ask how the *mix* of failure types changes as the horizon grows.

A.2 Training, Planning, and Optimization

Agents are trained with reinforcement learning (Ouyang et al., 2022), hierarchical planning (Liu et al., 2023a; Sun et al., 2023), tool use and reflection (Yao et al., 2023b; Shinn et al., 2023), and deliberation-style prompting (Wei et al., 2022; Yao et al., 2023a). Recent papers also study planning on its own, for example plan–act pipelines for web agents (Erdogan et al., 2025) and reliability of long plans in constrained settings (Chen et al., 2024). That work improves how agents act. We study which failure modes become dominant as horizons grow, and what that implies for design (Section 5).

A.3 Memory and Long-Context Reasoning

Long context windows do not by themselves fix long-range reasoning (Liu et al., 2023b). Better hardware and attention implementations (Press et al., 2022; Dao et al., 2022) sit alongside "lost in the middle" and similar issues. Retrieval and external memory (Madaan et al., 2022; Lewis et al., 2020; Park et al., 2023), compression for trajectories (Kang et al., 2025), and learned memory–reasoning stacks (Zhou et al., 2025) tackle scaling from the architecture side. Embodied setups also maintain explicit state (Wang et al., 2023; Ahn et al., 2022). We do not introduce a new memory module; we report how memory-related failures show up in labeled trajectories when we extend the horizon.

A.4 Failure analysis and diagnostics

Existing failure studies largely focus on domain-specific issues, such as code-generation errors (Chen et al., 2023), web interaction failures (Koh et al., 2024), and embodied stochasticity (Zeng et al., 2020). Broader agent benchmarks can provide coarse failure categories: AgentBench (Liu et al., 2023c), for example, reports high-level tags such as hallucination and planning errors, but does not study how failure patterns shift as task horizon increases. Recent taxonomy-oriented work on multi-agent LLM systems (Cemri et al., 2025) offers complementary diagnostics, yet its target setting is multi-agent coordination rather than single-agent, horizon-conditioned failures across domains. We instead develop a cross-domain taxonomy grounded in FMEA and a structured literature review (Appendix B), and validate trajectory-level labels with humans and a trajectory-grounded LLM-as-a-Judge pipeline (inter-annotator κ=0.61, human–judge κ=0.84; Appendix F).

B Appendix: Literature-to-Taxonomy Mapping

We systematically reviewed long-horizon agent literature using a shared spreadsheet (`Literature Review_Long-Horizon Reasoning in LLM agents.xlsx`). For each paper we recorded title, venue, domains, models, and free-text notes on reported failure mechanisms (the spreadsheet column Failure Modes). We then aligned recurring mechanisms to our seven-category taxonomy during iterative coding. Table 2 lists three representative prior works per category drawn from that review corpus and already present in our bibliography; these are illustrative anchors rather than an exhaustive list.

Table 2: *Representative prior work mapped to each failure category.* Entries are drawn from our literature spreadsheet and selected for cross-domain coverage; full paper lists and extracted notes remain in the supplementary spreadsheet.

Category	**Representative prior works (3)**	**Typical mechanism (from review notes)**
Environment	(Zhou et al., 2024a; Chen et al., 2025b; Koh et al., 2024)	Environment fidelity, transitions, or multimodal observation mismatch.
Instruction	(Trivedi et al., 2024; Chen et al., 2024; Wang et al., 2025b)	Ambiguous goals, missing constraints, or mis-specified task requirements.
Catastrophic Forgetting	(Sinha et al., 2025; Li et al., 2024; Feng et al.)	Loss of long-range constraints or competencies as trajectories lengthen.
False Assumption	(Zhang et al., 2025b; Yang et al., 2024; Wei et al., 2025)	Hallucinated facts, unfounded shortcuts, or misaligned beliefs about state.
Planning Error	(Erdogan et al., 2025; Si et al., 2025; Gonzalez-Pumariega et al., 2025)	Subgoal decomposition, ordering, or plan repair under dynamics.
History Error Accumulation	(Wang et al., 2025c; Liu et al., 2023c; Xi et al., 2025b)	Multi-step error propagation, unstable rollouts, or interaction loops.
Memory Limitation	(Kang et al., 2025; Zhou et al., 2025; Anwar et al., 2025)	Context limits, compression trade-offs, or long-horizon memory bottlenecks.

C Appendix: Failure Taxonomy Definitions

C.1 Environment Error

1. **Environment Disturbance / Problem Itself**

   Environment failures arise when **the external world in which an agent operates changes, degrades, or behaves stochastically in ways that are not explicitly modeled by the agent's plan or assumptions.** These disturbances include network or server instability, delayed or partial UI rendering, unexpected pop-ups or overlays, schema and data drift, protocol mismatches, and stochastic physical or simulator dynamics. Even when the agent's reasoning and action selection are locally correct, such environment-level disruptions can invalidate trajectories by altering the state, interface, or action semantics mid-execution.

   In long-horizon tasks, agents interact with the environment over extended time spans, increasing the probability that the environment changes between planning, perception, and execution. Cached assumptions (e.g., UI state, database schema, metrics, or action protocols) become stale, and small disturbances compound across steps, leading to cascading failures rather than immediate errors. As a result, long-horizon agents are particularly vulnerable unless they incorporate robust environment re-validation, recovery policies (retry, refresh, backoff), and explicit mechanisms to detect and adapt to environmental drift and noise.

2. **Unable to Detect Environment Change (limitations within the agent's interaction mechanisms)**

   This failure occurs when an **agent cannot reliably perceive, verify, or internalize a change (or non-change) in the environment, so its internal belief state diverges from the actual state.** It often manifests as false transitions (believing a navigation or command succeeded when it did not) or missed transitions (failing to notice the UI/OS/physical state has changed), driven by interaction bottlenecks such as limited observability, imperfect state parsing, unsupported action primitives (e.g., drag), or inability to handle obstacles like CAPTCHAs, banners, and overlays. The result is that subsequent actions are executed against the wrong target state (e.g., typing into the wrong page, issuing follow-up shell commands after a failed command, or proceeding in manipulation despite a misaligned object placement).

   Long-horizon tasks amplify this failure because state tracking must remain consistent across many steps, and even a single undetected mismatch early on can cascade into repeated invalid actions and compounding errors. Over longer trajectories, environments also present more opportunities for partial loads, transient failures, UI changes, and evaluation-sensitive deviations (e.g., "almost correct" placements), increasing the chance that the agent's interaction mechanism fails to register what actually happened. Without frequent re- observation and state re-validation, the agent's plan becomes increasingly decoupled from reality, making recovery progressively harder.

C.1.1 Web

This failure occurs when a web agent operates under non-stationary or imperfectly perceived environments, including stochastic network conditions, dynamic or partially loaded pages, pop-ups/CAPTCHAs, or mismatches between the agent's belief about the UI state and the actual rendered interface. In long-horizon web tasks, prolonged interaction increases the likelihood of such disturbances and belief–observation misalignment, causing agents to act on incomplete, outdated, or falsely assumed page states.

C.1.2 OS

This failure occurs when an OS agent operates under unexpected system-level conditions or misperceives command outcomes, such as permission denials, silent failures, or environment state changes that are not correctly reflected in the agent's belief state. In long-horizon OS tasks, repeated command execution increases the likelihood that unobserved failures accumulate and desynchronize the agent's internal state from the actual system state.

C.1.3 Embodied

This failure occurs when an embodied agent's execution is disrupted by external stochasticity or physical/environmental noise that invalidates previously correct plans or actions, or when the agent

fails to detect such changes in the physical state. In long-horizon embodied tasks, prolonged interaction increases exposure to randomness (e.g., slippage, drops, perturbations), making environment-induced misalignment increasingly likely.

C.1.4 Database

This failure occurs when a database agent operates under non-stationary system environments, such as schema drift, model version updates, telemetry format changes, or external processes modifying the database state without the agent's awareness. In long-horizon DB workflows, delayed execution and cached assumptions make agents especially vulnerable to acting on stale schemas or corrupted metrics.

C.2 Instruction Error

1. **Ill-defined Instruction**

   Ill-defined instruction failures occur when **the task specification is ambiguous, underspecified, internally inconsistent, or misaligned with the capabilities and constraints of the operating environment, making the intended goal state unclear or unreachable.** This includes unachievable goals (e.g., requesting non-existent fields or actions), out-of-scope follow-up queries where required information is missing, and implicit or vague constraints ("safe," "active users," "churn") that are not formally defined. In such cases, agents may hallucinate missing details, adopt incorrect default interpretations, or violate negative or safety constraints while believing they are following the instruction.

   Long-horizon tasks decompose high-level goals into many interdependent sub-instructions, multiplying opportunities for ambiguity, drift, and constraint loss as execution unfolds. Early misunderstandings or failed sub-steps propagate forward, causing later actions to be built on invalid assumptions rather than triggering clarification or replanning. As trajectories grow longer, the gap between the user's original intent and the agent's evolving interpretation widens, making ill-defined instructions a dominant source of cascading, hard-to-detect failures.

2. **Partially Understand Instruction**

   This failure occurs when **an agent grasps the high-level intent or primary action of an instruction but fails to fully internalize critical constraints, modifiers, exceptions, or conditional clauses.** As a result, the agent performs actions that are syntactically valid and seemingly reasonable, yet semantically incorrect with respect to the full instruction (e.g., ignoring "except," "only if," or key qualifiers like "mismatching"). The agent may also misread interaction feedback, prematurely concluding success despite visible error signals, warnings, or unmet conditions in the execution trace.

   In long-horizon tasks, instructions are often decomposed and re-represented across many steps, increasing the risk that fine-grained constraints are dropped, weakened, or forgotten over time. Partial understanding early in the trajectory can appear locally successful, masking the error until much later when accumulated effects surface and recovery becomes costly. Because long trajectories reward forward progress, agents may over-trust partial completion signals and continue execution without re-checking whether all instruction conditions were actually satisfied.

C.2.1 Web

This failure arises when a web agent is given ill-defined, unachievable, or out-of-scope instructions that are inconsistent with the actual web environment or lack sufficient information to determine a valid goal state. In long-horizon workflows, instructions are decomposed into many dependent sub-requests, increasing the chance that ambiguities or unreachable goals propagate forward and cause cascading failures.

C.2.2 OS

Instruction failures arise when an OS agent understands the general task but misses critical constraints, exceptions, or negative instructions embedded in the prompt. In long-horizon settings, such omissions often propagate across multiple shell operations, leading to globally incorrect system modifications.

C.2.3 Embodied

Instruction failures arise when an embodied agent captures only part of a task description but ignores critical modifiers or constraints, leading to correct actions applied to incorrect targets. Such partial

understanding becomes more damaging in long-horizon tasks where early misinterpretations propagate across many physical steps.

C.2.4 Database

Instruction failures arise when a database agent misinterprets ambiguous, underspecified, or context-dependent natural-language requests, failing to align them with the correct business semantics or safety constraints. In long-horizon interactions, such misunderstandings propagate across planning and execution phases.

C.3 Catastrophic Forgetting

Catastrophic forgetting refers to the agent's **failure to retain and consistently apply previously acquired knowledge, constraints, or decisions while continuing to operate or adapt over time.** This includes forgetting explicit user instructions (e.g., "never modify production" or "exclude the payments schema"), earlier clarifications, rejected hypotheses, or prerequisite skills needed for later subtasks. The failure can arise from limited context windows, inadequate memory retrieval, or interference between newly introduced information and earlier task- relevant state, causing the agent to revert to generic assumptions or unsafe defaults.

Long-horizon tasks inherently exceed the agent's short-term context capacity and require stable knowledge persistence across many turns, phases, or sessions. As execution progresses, early constraints and learned insights are more likely to scroll out of context or be overshadowed by recent interactions, leading to overwriting rather than accumulation of knowledge. This results in cascading errors, repeated mistakes, and breakdowns in dependency chains, where later steps silently violate earlier commitments or prerequisites that are no longer actively represented.

C.3.1 Web

Catastrophic forgetting occurs when a web agent loses or stops enforcing earlier task-critical constraints (e.g., "new only," "no spoilers," "use main branch") as the interaction progresses, often because those constraints drift out of active context or are overwritten by later subgoals.

C.3.2 OS

Catastrophic forgetting occurs when an OS agent loses earlier task constraints, safety requirements, or intermediate conclusions over the course of a multi-step execution. As OS tasks grow longer, earlier safeguards can drift out of active context, leading to irreversible or unsafe actions.

C.3.3 Embodied

Catastrophic forgetting occurs when an embodied agent fails to retain prerequisite skills or learned environmental knowledge required for later stages of a long task. This is especially pronounced in hierarchical or curriculum-like tasks where later success depends on reliably transferring earlier competencies.

C.3.4 Database

Catastrophic forgetting occurs when a database agent loses earlier constraints, preferences, or hypotheses over extended multi-turn workflows, even though they remain relevant to later decisions. This leads to unsafe or inconsistent actions as long-horizon execution exceeds the agent's effective retention capacity.

C.4 False Assumption

1. **Universal Facts (Overgeneralized World Knowledge)**

   This failure occurs when an **agent relies on implicit assumptions, heuristics, or patterns learned from prior experience and treats them as universally true, without verifying whether they hold in the current task or environment.** Unlike environment failures—where the external world changes or behaves unpredictably—this error arises from the agent's incorrect prior belief about how the environment should behave, leading it to generate actions or reasoning based on a false premise. These false assumptions can include hallucinated APIs or schema elements, default interpretations of goals, over-trusting probabilistic components as deterministic, or inferring causality from correlation. Even when local reasoning steps appear coherent, the agent's trajectory is grounded in an incorrect premise, causing repeated ineffective actions or confidently wrong decisions.

   Long-horizon reasoning encourages agents to commit early to implicit assumptions and reuse them across many steps, making it less likely that those assumptions are revisited or challenged later. As trajectories grow longer, confirmation bias sets in: intermediate outputs are interpreted as validation rather than signals to re-check premises, allowing false

assumptions to silently propagate and compound. Without explicit mechanisms for assumption verification or hypothesis revision, long-horizon agents increasingly drift from the true environment dynamics while remaining internally consistent.

2. **Long- Horizon Complex Task Reasoning**

   This failure arises when an agent **implicitly assumes it can sustain correct multi-step logic, control flow, and state consistency across long and complex task sequences, but in practice loses coherence as execution unfolds.** It manifests as incorrect loop or conditional logic, premature termination, infinite repetition, or divergence mid-trajectory due to unverified assumptions about system state (e.g., file ownership, working directory, or intermediate progress). Although individual actions may be locally reasonable, the agent's global reasoning chain breaks because earlier assumptions are neither validated nor revised.

   Long-horizon tasks require maintaining structured plans, intermediate invariants, and progress tracking over many dependent steps, which standard agents often approximate with shallow or implicit reasoning. As task length increases, small logical errors or unchecked state assumptions compound, causing the agent to drift off-goal or exhaust interaction budgets without recovery. Without explicit planning, loop verification, and state re-checking, agents increasingly rely on false assumptions of coherence, making complex long-horizon tasks a primary failure regime.

C.4.1 Web

This failure occurs when a web agent fills in missing information or applies generic priors as if they were universally valid, instead of grounding decisions in what is actually observable on the webpage (e.g., fabricating an email, assuming a policy is allowed, or treating ambiguity as resolved).

C.4.2 OS

This failure occurs when an OS agent assumes system properties or command semantics without verification, relying on generic priors instead of observed system state. Such assumptions can lead to incorrect branching decisions or invalid operations, even when commands are syntactically correct.

C.4.3 Embodied

This failure occurs when an embodied agent overgeneralizes from common training patterns, assuming that a visually similar scene implies a standard goal or action, regardless of the actual instruction. Such bias-driven assumptions override task-specific constraints and lead to systematically wrong behavior.

C.4.4 Database

This failure occurs when a database agent overgeneralizes schema patterns, operator semantics, or workload behavior, treating them as universally valid without verification. Such assumptions override actual database-specific details and lead to systematically incorrect reasoning.

C.5 Planning Error

1. **Sub-plan.** Sub-plan planning errors occur when an **agent mis-decomposes a high-level goal into incorrect, incomplete, poorly ordered, or inefficient sub-steps, even though the overall objective is understood.** This includes choosing the wrong API or parameter format for a specific step, omitting prerequisite actions, violating constraints within a sub-plan, or adopting an inefficient strategy that exhausts time, memory, or interaction budgets. The failure is localized at the sub-plan level, but it breaks the global task by introducing deadlocks, irreversible actions, or cascading inefficiencies.

   Long-horizon tasks magnify sub-plan errors because later steps critically depend on the correctness and ordering of earlier ones, and recovery options shrink as execution progresses. A single flawed sub-plan (e.g., dropping backups too early, rebuilding indexes before diagnosis) can invalidate all downstream actions or make rollback difficult. As plans grow deeper and more branched, agents without explicit plan validation or cost-aware reasoning increasingly commit to locally plausible but globally harmful sub-plans, leading to systematic long-horizon failure.

2. **Action.** Action-level planning errors occur when an **agent selects or executes an incorrect concrete action, despite having an appropriate high-level plan or sub-task in place.** These failures include execution

crashes (e.g., syntax or type errors), semantically invalid actions that do not achieve the intended effect in the environment, or mismatches between planned intent and low-level action realization (e.g., incorrect API flags, shell commands, or motor control tokens). As a result, the action either fails outright or silently alters the environment in unintended ways.

In long-horizon tasks, agents must execute many interdependent actions, making even small action-level errors costly because they propagate forward and constrain future options. Repeated low-level failures can exhaust interaction budgets, corrupt intermediate state, or trigger compounding side effects before the agent detects the problem. Without robust action validation, rollback, and feedback-aware correction, long-horizon execution amplifies these errors from isolated glitches into trajectory-ending failures.

C.5.1 Web

Planning errors occur when a web agent chooses an incorrect sequence of subgoals or misses key prerequisite steps, leading it to pursue the wrong navigation branch, ignore constraints, or stop prematurely even if the end goal is clear.

C.5.2 OS

Planning errors arise when an OS agent fails to decompose a high-level instruction into a correct and ordered sequence of shell operations, leading to missing steps, incorrect ordering, or inappropriate operator selection. These errors often surface before execution, but their effects compound over long execution chains.

C.5.3 Embodied

Planning errors arise when an embodied agent generates an incorrect sequence of sub-goals or fails to integrate multi-step logical requirements, even though individual actions may be physically executable. In long-horizon settings, weak decomposition or missing logical dependencies leads to deadlocks or goal divergence.

C.5.4 Database

Planning errors arise when a database agent constructs an incorrect or suboptimal sequence of logical or operational steps, even if each individual operation is valid in isolation. In long-horizon DB tasks, poor plan ordering or granularity can cause irreversible damage or severe inefficiency.

C.6 History Error Accumulation

History error accumulation refers to failures where **small mistakes, incorrect assumptions, or misspecified intermediate artifacts introduced early in a trajectory persist in the agent's working state and progressively distort downstream reasoning and actions**. The "history" can include intermediate tables, cached schemas, partial code edits, stored hypotheses, retrieved documents, or planning traces. Because later steps reuse these artifacts as if they were correct, the system can produce outputs that are internally consistent yet fundamentally wrong (e.g., subtle over-counting, reasoning over the wrong feature column, or refining a plan built on a misdiagnosis).

Long-horizon workflows amplify this failure because they involve many dependent iterations, where each step inherits context and intermediate products from previous steps. Even a minor upstream error (a missing filter, wrong ID, biased retrieval) can silently propagate across multiple tools, agents, and refinements, making the final failure difficult to localize and correct. As trajectories lengthen and artifacts become more complex (e.g., long SQL scripts or multi-stage playbooks), agents increasingly apply "patchy" local edits instead of globally re-validating assumptions, allowing errors to compound into trajectory-level breakdowns.

Annotation scope. History Error Accumulation is reserved for the *compounding pattern itself*, as a trajectory-level phenomenon. It does *not* require any prior step to be labeled with another error type; it applies when no single step contains a label-worthy error sufficient to cause the failure on its own, yet the trajectory fails because individually-minor drifts compound across many steps. When a single labelable error at one step is the proximate cause of the failure, that step's category takes precedence and downstream consequences are not separately labeled as History Error Accumulation.

C.6.1 Web

History error accumulation occurs when a web agent fails to correctly interpret or update its interaction history, causing small mistakes or ineffective actions to be repeated and compounded over time rather than corrected. In long-horizon web tasks, such uncorrected repetitions (e.g., reusing stale element IDs, appending corrupted queries, or

toggling controls) can exponentially degrade the agent's state and exhaust the interaction budget.

C.6.2 OS

History error accumulation occurs when errors or misinterpretations in early command executions are not corrected, and the agent continues reasoning as if those steps succeeded. Over long OS interaction trajectories, such uncorrected assumptions compound and increasingly diverge from the true system state.

C.6.3 Embodied

History error accumulation occurs when an embodied agent fails to correct an early planning or execution error and continues acting based on that flawed history, compounding the failure over time. In long-horizon tasks, this often results in persistent loops or deadlocks instead of replanning.

C.6.4 Database

History error accumulation occurs when early logical or data-level mistakes are reused by downstream steps, causing subtle errors to compound across queries, plans, or optimizations. Because later stages treat earlier artifacts as ground truth, the system becomes increasingly confident yet increasingly wrong.

C.7 Memory Limitations

Memory limitation failures occur when an agent cannot retain, retrieve, or reason over all task-relevant information because **the interaction history, environment state, or intermediate artifacts exceed its effective memory capacity**. This includes exceeding the model's context window, lossy summarization of prior steps, or selective retrieval that omits critical constraints, schema elements, or past observations. As a result, the agent may operate with an incomplete or distorted view of the task, leading to inconsistent joins, dropped filters, unsafe recommendations, or misinterpretation of system state.

Long-horizon tasks inherently accumulate large amounts of state—schemas, logs, plans, hypotheses, and prior decisions—quickly overwhelming fixed-context or shallow memory mechanisms. As execution continues, earlier but still critical facts fall out of context or are compressed into coarse summaries, making subtle edge cases and historical dependencies invisible to later reasoning. Without robust external memory, selective retrieval, and consistency checks, long-horizon agents increasingly behave as if earlier constraints or observations never existed, causing systematic drift and brittle decision-making.

C.7.1 Web

Memory limitation failures arise when a web agent cannot retain and compare multi-item evidence across steps (due to context limits or lack of external scratchpad), causing it to drop earlier observations, constraints, or candidate details.

C.7.2 OS

Memory limitation failures arise when an OS agent cannot retain earlier command outputs, constraints, or environment details because the interaction history exceeds its effective context capacity. In long OS tasks, this leads to violations of earlier constraints or repeated mistakes due to missing historical information.

C.7.3 Embodied

Memory limitation failures arise when an embodied agent cannot maintain or retrieve sufficient state information over extended horizons, such as object locations, intermediate goals, or past outcomes. As task length increases, limited internal memory leads to inconsistent behavior and loss of progress tracking.

C.7.4 Database

Memory limitation failures arise when a database agent cannot keep large schemas, long interaction histories, or complex multi-modal plans fully in context, forcing lossy summarization or selective retrieval. Over long horizons, this results in dropped constraints, missing joins, or inconsistent reasoning across turns.

C.8 Representative Failure Cases

Table 3: *Representative failure cases across domains and failure categories.* Each row corresponds to a failure category in our taxonomy, and each column shows a concrete example from a different domain—Web, OS, Embodied, and Database. **[L]** denotes failure categories that are predominantly long-horizon-specific (rare in short-horizon settings), while **[S]** denotes categories observed in both short- and long-horizon tasks but often amplified under longer horizons. Details on taxonomy development are provided in Section 3.2, and additional examples and explanations are provided in Appendix C.

Category	Web	OS	Embodied	DB
Environment [S] (Environment Disturbance & Unable to Detect Environment Change)	After a slow page load with a transient pop-up banner, the agent *assumes the page navigation has completed* and believes it is on the profile page ("Mark Johnson"), while the UI is actually still on a different profile ("mike_chen"), resulting in actions applied to an *incorrect environment state*.	A shell command fails due to *permission denial*, but the agent *fails to detect the error signal and believes the environment state has changed*, continuing to issue follow-up commands that depend on the nonexistent state change.	In a simulator with *stochastic dynamics*, a robot repeatedly drops a block due to random gripper failure, but the agent *fails to re-observe the changed environment state*, causing repeated execution failures despite correct high-level reasoning.	After a *database schema change* renames columns or adds tables, the agent continues generating SQL based on an outdated schema snapshot because it *fails to re-check the current schema*, producing invalid queries or silently incorrect results.
Instruction [S] (Ill-defined Instruction & Partially Understand Instruction)	An agent is asked to retrieve a specific product field that *is not defined on the target website*, meaning the *instruction requests a non-existent attribute*. Instead of detecting the mismatch and asking for clarification, the agent fabricates a plausible value and proceeds.	The agent is instructed to change file permissions "except those of mine," where the *instruction contains an ambiguous ownership constraint*. The agent misinterprets it and applies a recursive `chmod` globally.	Given the instruction "put the block in the mismatching bowl," the agent grasps the block but *fails to interpret the key constraint "mismatching"* and places it into a matching-colored bowl.	When asked to "show churned users," the *instruction leaves the definition of churn unspecified*. The agent therefore applies a naïve definition (no login in 30 days) instead of the organization's actual churn definition.
False Assumption [S]	When a contact form requires identity fields and the instruction says "do not invent personal info," the agent still enters a made-up name and email because it *assumes placeholder personal information is acceptable*.	The agent *assumes certain files are owned by the user* and excludes them from permission changes without checking ownership metadata.	In a mismatching-bowl task, the agent *assumes the goal is the common "matching" variant* and places a red block into a red bowl despite the instruction requiring a mismatching placement.	A SQL-generation agent *assumes every table contains a* `created_at` *column* and repeatedly invents similar column names in `WHERE` clauses.
Planning Error [S] (Sub-plan & Action)	While planning the next step to find the highest-upvoted answer, the agent *fails to switch sorting from "New" to "Top"*, then cites a recently seen low-upvote comment as the "highest-upvoted" evidence.	When planning how to update permissions, the agent *chooses an incorrect action plan and directly applies a blanket* `chmod` instead of first identifying eligible files.	While decomposing the task into steps, the agent *generates an incorrect sub-plan and attempts to stack blocks before clearing the base area*, producing an invalid task order.	During database migration planning, the agent *executes an incorrect action sequence and drops the backup before completing data transfer*, leaving no recovery path when the migration fails.

(continued on next page)

Table 3 (continued)

Category	Web	OS	Embodied	DB
Catastrophic Forgetting [L]	The agent initially applies the "Condition: New" filter but later adds a "Renewed" item to the cart after a long navigation sequence because the *"new-only" constraint is still present in the context but not attended to during later reasoning*.	Early in a task, the agent is instructed not to modify sensitive system files, but later generates and executes a script that deletes `/root/.bashrc` even though the *original safety instruction is still in the context but ignored during execution*.	In Minecraft, an agent that has learned to craft wood and stone tools fails on advanced "Iron+" tasks because it *does not reuse earlier knowledge that is still present in the context*, leading to failure in later crafting stages.	After the user specifies "never touch the payments schema," a later autonomous cleanup step proposes dropping an index on the payments tables because the *constraint remains in the interaction history but is overlooked during later planning*.
History Error Accumulation [L]	After an element click fails, the agent *repeats the same failed action across multiple steps*, allowing a *small initial mistake to accumulate into repeated ineffective actions*.	After issuing an invalid command that fails silently, the agent *treats the failed step as successful* and continues executing dependent commands, causing *the early incorrect state to propagate through the workflow*.	After selecting an incorrect initial sub-task, the agent *continues executing downstream steps based on that incorrect earlier decision*, gradually *amplifying the initial mistake*.	An early intermediate table *omits the filter* `is_deleted = false`, and all downstream analytics reuse that table, causing *the small upstream error to compound into systematically inflated results*.
Memory Limitation [L]	While scrolling through a contributors list, the agent loses the earlier commit counts because *the interaction history exceeds the context window*, forcing the system to *drop earlier observations*.	Earlier notices about file ownership are lost because *earlier messages are truncated from the context window* during a long interaction session.	In long-horizon Minecraft tasks, earlier navigation decisions are lost because *the trajectory exceeds memory capacity and earlier steps are summarized, causing critical information to be dropped from context*.	In a warehouse with hundreds of tables, the agent retains only part of the schema because *memory retrieval returns an incomplete schema snapshot*.

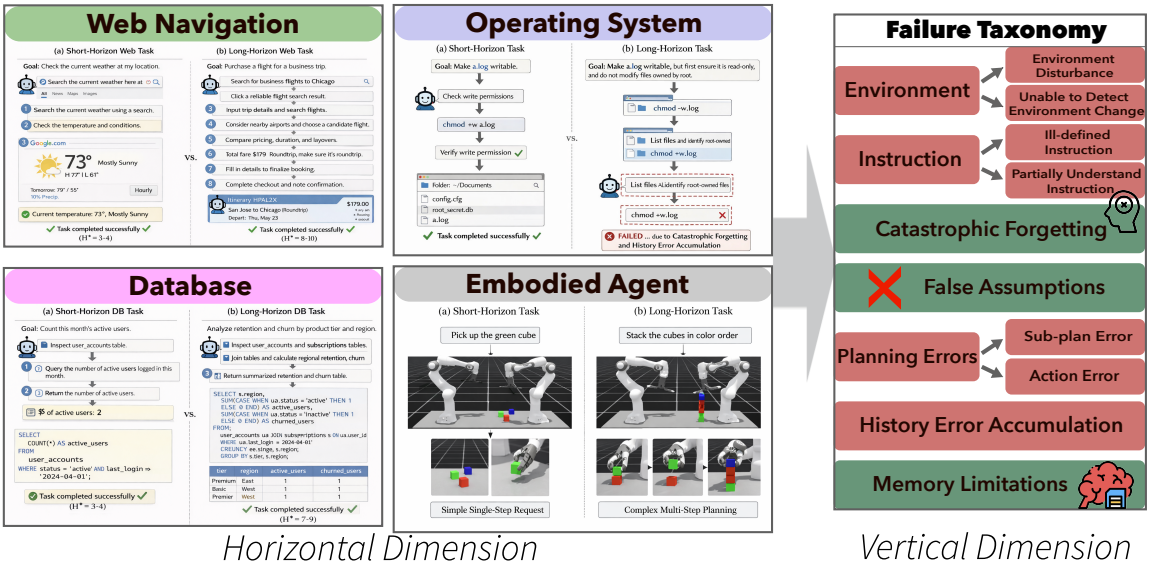

Figure 5: HORIZON *overview with two orthogonal dimensions. Left* (Horizontal / Horizon): four domain examples contrasting short- vs. long-horizon tasks under our task-structure definition, where intrinsic horizon H^* increases with extension level s (and may also increase compositional depth C). From top-left to bottom-right: Web illustrates theoretical H^* growth (e.g., "purchase a flight," $H^* = 8$, vs. "check the weather," $H^* = 3$); OS shows depth extension by inserting non-skippable intermediate states (e.g., enforce and verify read-only before making `a.log` writable); Database and Embodied show breadth/length extension by composing multiple short-horizon subtasks into one workflow, increasing number of subtasks and coordination steps. *Right* (Failure Attribution): 7-category failure taxonomy. Colors indicate a hierarchical risk perspective inspired by FMEA: blue corresponds to design-level risks (DFMEA), originating from limitations in agent architecture, while orange corresponds to process-level risks (PFMEA), emerging during sequential execution. More details are provided in Section 3.

D Appendix: Experiment 1 (Cross-Domain Failure Analysis) Set-up

D.1 Data, Environments, and Agents

Given the novelty of our horizon definition and diagnostic framework, no existing benchmark directly supports the proposed notion of intrinsic horizon and controlled extension. We therefore construct our evaluation data using the HORIZON workflow illustrated in Figure 1. Across all domains, baseline tasks are first verified to be solvable, ensuring that performance degradation arises from increased horizon rather than task ambiguity or agent misconfiguration.

1. *Web Navigation*. We adopt WebArena (Zhou et al., 2024a) as the base environment. From the benchmark task pool, we select tasks that achieve roughly 90% success under a baseline run with GPT-5-mini, and apply our horizon extension procedures to construct task families with increasing intrinsic horizon.
2. *Operating Systems (OS)*. We build on OS tasks from AgentBench (Liu et al., 2023c), again selecting tasks with perfect baseline performance under GPT-5-mini before applying depth-based horizon extension.
3. *Databases (DB)*. We build on MAC-SQL (Wang et al., 2025a), a multi-agent Text-to-SQL framework, to support controlled query decomposition and task composition. The agent follows a modular pipeline consisting of three components: a *Selector*, a *Decomposer*, and a *Refiner*. Given a user query, the Selector first identifies relevant tables and schema elements. The Decomposer then breaks the query into sub-questions and generates intermediate SQL statements, which are composed into a final query. Finally, the Refiner executes the generated SQL, detects errors (e.g., execution or syntax errors), and iteratively revises the query to produce the final result. We apply depth-based horizon extension.
4. *Embodied.* To evaluate variable-horizon tasks in an embodied setting, we build a bimanual robot-arm simulation environment in IsaacSim 5.0 (Kachaev et al., 2025). The environment contains two Franka Emika Panda robot arms, each equipped with a Tesollo DG-3F-B three-finger gripper, and is populated with three cubes (red, blue, green). The agent controls the arms through four parameterized primitive actions, summarized in Table 4. For each task, the agent receives the available actions and object locations as part of the task context, and outputs a JSON plan over the four primitives. The plan is then executed in simulation, and task success is judged manually: a task is marked successful if and only if the target object configuration (e.g., "red cube stacked on blue cube") is achieved at the end of execution, with no objects dropped or displaced outside the workspace. We apply breadth-based horizon extension.

Action	Parameters	Effect
`home`	—	Reset both arms to default pose.
`move_to_pose`	`arm`, `pose`	Move `arm` to target Cartesian pose.
`grasp`	`arm`	Close gripper on `arm`.
`release`	`arm`	Open gripper on `arm`.

Table 4: Primitive actions exposed to the agent in the embodied environment. `arm` $\in$ {`left`, `right`}; `pose` specifies position and orientation.

D.2 Models

To demonstrate feasibility on current frontier systems, we evaluate two state-of-the-art representative model families. We use GPT-5-mini (OpenAI, 2025b) and Claude-4-Sonnet (Anthropic, 2025) across all four domains. We additionally evaluate the full GPT-5 (OpenAI, 2025a) on the embodied domain to confirm the same degradation pattern.

D.3 Evaluation

We evaluate agent performance as a function of the HORIZON extension level s, reporting success rates over task sets with identical s. In addition to outcome-level success, failed trajectories are analyzed using our seven-category failure taxonomy (Table 1) through a combination of expert human annotation and calibrated LLM-based judges.

To quantify variability, we report mean and standard deviation over three independent runs for each horizon setting. In the Embodied domain, both model families show clear horizon-dependent degradation, with higher variance at intermediate horizons. For GPT-5 variants, success decreases from 0.926 ± 0.064 at short horizon to 0.407 ± 0.170, 0.259 ± 0.231, and 0.074 ± 0.064 as horizon increases. Claude-4-Sonnet shows a similar trend, dropping from 0.963 ± 0.064 to 0.259 ± 0.064, 0.111 ± 0.111, and 0.037 ± 0.064.

We also emphasize that the extension level s measures compositional depth (the number of high-level subtasks composed within a task), not raw action count: each unit of s expands into multiple low-level actions during execution, so the action-level intrinsic horizon $H^*(s)$ grows substantially faster than s. Concretely, per-trajectory action budgets in our evaluation reach up to 150 (Web), 50 (OS), 31 (Database), and 34 (Embodied) steps, comparable to or exceeding the effective horizons reported in recent long-horizon agent benchmarks.

D.4 Reproducibility and Responsible Use

The benchmarks we build on (WebArena, AgentBench, MAC-SQL, Isaac Sim) are released under permissive research licenses, and our use is consistent with their intended research purpose. HORIZON task families, annotated trajectories, and judge prompts will be released under CC BY-NC 4.0 for non-commercial research use. All tasks are synthetic or drawn from public agent benchmarks and contain no personally identifying information or offensive content.

All experiments use the official APIs of GPT-5-mini, GPT-5, and Claude-4-Sonnet; parameter counts are not publicly disclosed by the providers. We use default decoding settings (temperature $= 1.0$) unless otherwise noted. We do not perform hyperparameter search. Exact API versions, benchmark commits, and random seeds will be released with the code.

Failure annotation was performed by two of the authors with prior research experience on LLM agents, using the decision tree in Figure 8 and the per-domain definitions in Table 1. No external annotators were recruited and no monetary compensation was involved.

D.5 Additional Results

See Figure 6 and Figure 7.

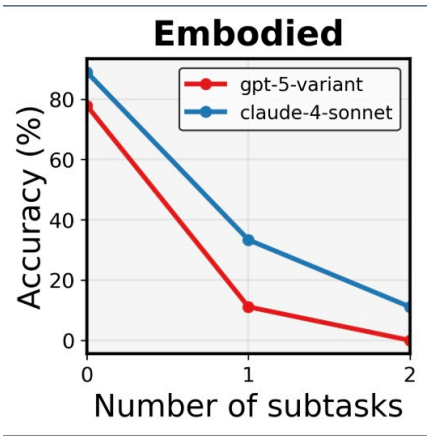

Figure 6: Embodied task sets with GPT-5.

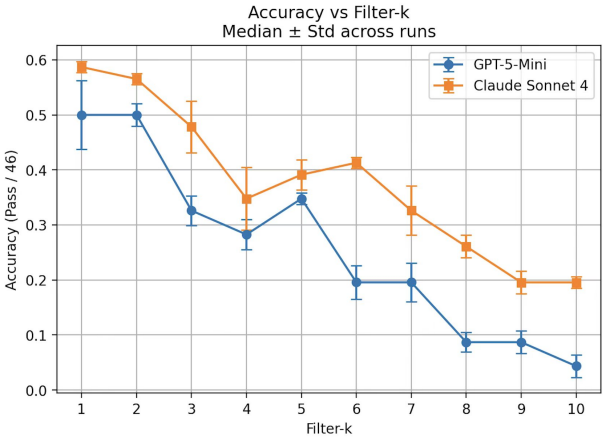

Figure 7: Unfiltered task sets in the OS domains.

E Appendix: 5-domain × 7-type Failure Mini-Trajectories

E.1 Web

Environment

Step 1 (FAILURE STEP)
Summary: The agent attempts to go to the homepage and issues a goto that resolves to localhost:4399.
Raw Trace:

```
Page : Tab 0 (current): One Stop Market
Agent: I need to analyze this objective which has three parts:
1. What do customers say about brush from sephora
2. Tell me the total cost of my latest cancelled order
3. Tell me the total cost of my latest pending order

I'm currently on the One Stop Market homepage, but I need to go to
Sephora for the first part of the objective. Let me navigate to the
homepage to find Sephora.

In summary, the next action I will perform is
goto [http://homepage.com]

Action: goto [localhost:4399]
```

Step 2
Summary: Seeing a Chrome error page, the agent retries navigating to the homepage but again executes goto localhost:4399.
Raw Trace:

```
Page : Tab 0 (current):
Agent: I can see that the previous action tried to navigate to
localhost:4399 but resulted in a Chrome error page. This suggests
the local server might not be running or the URL is incorrect.
Let me try to navigate to the homepage to find available websites.

In summary, the next action I will perform is
goto [http://homepage.com]

Action: goto [localhost:4399]
```

Step 3
Summary: The agent notes the error again and repeats the same navigation action to localhost:4399.
Raw Trace:

```
Page : Tab 0 (current):
Agent: I can see that the current page is showing a Chrome error
page, which means the URL I tried to navigate to (localhost:4399)
is not accessible. Let me try to navigate to the homepage to see
what websites are available.

In summary, the next action I will perform is
goto [http://homepage.com]

Action: goto [localhost:4399]
```

Instruction

Step 1
Summary: On the order page, the agent clicks the Cancel button.
Raw Trace:

```
[Step 8]
Page : Tab 0 (current): #000000307 / Orders / Operations / Sales / Magento Admin
Agent: In summary, the next action I will perform is
click [13783]

Action: click [13783] where [13783] is button 'Cancel'
```

Step 2
Summary: The agent confirms the cancellation by clicking OK.
Raw Trace:

```
[Step 9]
Page : Tab 0 (current): #000000307 / Orders / Operations / Sales / Magento Admin
Agent: In summary, the next action I will perform is
click [16102]

Action: click [16102] where [16102] is button 'OK' focused: True
```

Step 3 (FAILURE STEP)
Summary: The agent stops with a narrative message instead of the required single-word answer "Canceled".
Raw Trace:

```
[Step 10]
Page : Tab 0 (current): #000000307 / Orders / Operations / Sales / Magento Admin
Agent: In summary, the next action I will perform is
stop [Order #000000307 is already canceled]

Action: stop [Order #000000307 is already canceled]
```

1721

Catastrophic Forgetting

Step 1
Summary: The agent clicks the first product's "Add to Wish List" button from the laundry detergent results.
Raw Trace:

```
[Step 3]
Page : Tab 0 (current): Search results for: 'laundry detergent'
Agent: Let me add the first laundry detergent (Tide PODS)
to my wish list.

Action: click [5136] where [5136] is button 'Add to Wish List'
```

Step 2
Summary: On the Wish List page, the agent confirms the detergent was added and navigates to My Orders.
Raw Trace:

```
[Step 4]
Page : Tab 0 (current): My Wish List

Agent: The detergent has been added to the wish list.
Now I need to check my latest cancelled order.

Action: click [7019] where [7019] is link 'My Orders'
```

Step 3 (FAILURE STEP)
Summary: Despite reading the latest cancelled order and its total ($365.42), the agent wrongly decides it still needs to add a detergent.
Raw Trace:

```
[Step 5]
Page : Tab 0 (current): My Orders

Agent: The latest cancelled order is #000000170
with a total of $365.42.

Now I need to add a laundry detergent to my wish list.

Action: click [9227] menuitem 'Home & Kitchen'
```

Step 4
Summary: The agent navigates to Home & Kitchen and starts searching again.
Raw Trace:

```
[Step 6]
Page : Tab 0 (current): Home & Kitchen

Action: type [12303] [laundry detergent]
```

Step 5
Summary: The agent scrolls the search results to find products.
Raw Trace:

```
[Step 7]
Page : Tab 0 (current): Search results for: 'laundry detergent'

Action: scroll [down]
```

False Assumption

Step 1
Summary: On the Customer Reviews Report, the agent clicks "Show Reviews".
Raw Trace:

```
[Step 4]
Page : Customer Reviews Report / Magento Admin

Action: click [2068] link 'Show Reviews'
```

Step 2
Summary: The agent observes one customer with one review but switches to the "By Products" report.
Raw Trace:

```
[Step 5]
Page : Customer Reviews Report

Agent: Emma Lopez has 1 review.
Let me check the "By Products" report.

Action: click [1611] link 'By Products'
```

Step 3 (FAILURE STEP)
Summary: The agent misinterprets "126 records found" as the total number of reviews.
Raw Trace:

```
[Step 6]
Page : Product Reviews Report

Agent: The page shows "126 records found",
which indicates the total number of reviews.

Action: stop [126]
```

Planning Error

Step 1
Summary: From the Bestsellers report page, the agent opens the Customers section.
Raw Trace:

```
Page : Bestsellers Report / Magento Admin

Action: click [5882] link 'CUSTOMERS'
```

Step 2
Summary: The agent opens the "All Customers" page.
Raw Trace:

```
Page : Bestsellers Report

Action: click [6286] link 'All Customers'
```

Step 3 (FAILURE STEP)
Summary: Instead of displaying the customer grid, the agent attempts to export the customer list.
Raw Trace:

```
Page : Customers / Magento Admin

Agent: To show all customers, I will export the list.

Action: click [8519] button 'Export'
```

Step 4
Summary: The agent continues the export workflow.
Raw Trace:

```
Page : Customers / Magento Admin

Action: click [10946] button 'Export'
```

Step 5
Summary: The agent switches focus back to Reports, indicating a lack of coherent plan.
Raw Trace:

```
Page : Customers / Magento Admin

Action: click [6998] link 'REPORTS'
```

History Error Accumulation

Step 1
Summary: On the My Orders page, the agent prematurely navigates to the Video Games category instead of inspecting the orders to find the latest cancelled order total.
Raw Trace:

```
[Step 4]
Page : My Orders

Agent: I will open the Video Games category first.

Action: click [8961]  # Video Games
```

Step 2
Summary: After returning to the My Orders page later in the trajectory, the agent repeats the same navigation mistake and again leaves the page without checking the order information.
Raw Trace:

```
[Step 10]
Page : My Orders

Agent: I will open the Video Games category first.

Action: click [24391]  # Video Games
```

Step 3
Summary: The agent correctly identifies the cancelled order total ($365.42) but still navigates away instead of reporting the answer, allowing the earlier mistake to persist.
Raw Trace:

```
[Step 32]
Page : My Orders
```

```
Agent: The latest cancelled order visible here is
order 000000170 dated 5/17/23 with total $365.42.
I will report it after navigating to Video Games.

Action: click [82213]  # Video Games
```

Step 4 (FAILURE STEP)
Summary: Because the agent repeatedly leaves the relevant page instead of reporting the answer, the trajectory eventually reaches the maximum step limit and terminates without completing the task.
Raw Trace:

```
[Step 61]
Page : Video Games

Agent:

Action: stop [Early stop: Reach max steps 60]
```

History Error Accumulation

Step 1
Summary: The agent counts case-insensitive occurrences of the word "error" in all log files and writes the results into logs/file_error_counts.txt.
Raw Trace:

```
[Step 1]

Agent: Count case-insensitive "error" occurrences in
each log file and write "filename: count" to
logs/file_error_counts.txt.

Action: bash
awk 'BEGIN{IGNORECASE=1} {n=gsub(/error/,"&");
counts[FILENAME]+=n}
END{for (f in counts) print f": "counts[f]}' \
logs/log_*.txt > logs/file_error_counts.txt
```

Step 2
Summary: The agent checks the number of lines in the generated file and observes 8 entries instead of the expected 7.
Raw Trace:

```
[Step 2]

Agent: Check that logs/file_error_counts.txt contains
seven lines.

Action: bash
wc -l logs/file_error_counts.txt
```

Step 3
Summary: To diagnose the discrepancy, the agent inspects the contents of the output file.
Raw Trace:

```
[Step 3]

Agent: Inspect the file contents to understand why
there are 8 lines instead of 7.

Action: bash
cat logs/file_error_counts.txt
```

Step 4 (FAILURE STEP)
Summary: Instead of continuing the debugging process, the agent abruptly executes an unrelated command to count files in /etc, indicating that the earlier reasoning steps have gradually drifted away from the original task objective.
Raw Trace:

```
[Step 4]

Agent: Count the number of regular files directly
in /etc.

Action: bash
find /etc -maxdepth 1 -type f | wc -l
```

E.2 OS

Environment

Step 1
Summary: Make the wrapper script executable using chmod.
Raw Trace:

```
chmod +x date-format
(OS response: empty)
```

Step 2
Summary: Create README.md using a heredoc containing usage instructions.
Raw Trace:

```
cat > README.md << 'EOF'
# Date Format Tool
...
EOF
```

Step 3 (FAILURE STEP)
Summary: Attempt to list directory contents but receive no output.
Raw Trace:

```
ls -la
(OS response: empty)
```

Step 4
Summary: Try to print the working directory but again receive empty output.
Raw Trace:

```
pwd
(OS response: empty)
```

Step 5
Summary: Run a simple echo command which also returns no output.
Raw Trace:

```
echo "test"
(OS response: empty)
```

Instruction

Step 1
Summary: Count lines in verification.log to check for discrepancies.
Raw Trace:

```
wc -l verification.log
```

Step 2
Summary: Generate final_prep.txt reporting file and directory counts.
Raw Trace:

```
awk 'NR==1{f=$0} NR==2{d=$0}
END{printf "Verified file count: %s\nVerified directory count: %s\n",f,d}'
etc_counts.txt > final_prep.txt
```

Step 3 (FAILURE STEP)
Summary: Incorrectly answers using only the file count rather than the required total.
Raw Trace:

```
answer(There are 58 files in /etc.)
```

Catastrophic Forgetting

Step 9
Summary: Attempt a multiline awk script but quoting fails.
Raw Trace:

```
awk ' ... ' temp_file_list.txt > txt_sizes.csv
(OS response echoes script)
```

Step 10
Summary: Retry using a one-line awk command.
Raw Trace:

```
awk ' ... ' temp_file_list.txt > txt_sizes.csv
(OS response: empty)
```

Step 11 (FAILURE STEP)
Summary: Abruptly switches to counting files under /etc, abandoning the original task.
Raw Trace:

```
find /etc -type f | wc -l
-> 162
```

Step 12
Summary: Reports the /etc file count as the final answer.
Raw Trace:

```
answer(There are 162 regular files in /etc.)
```

False Assumption

Step 1
Summary: List regular files into file_list.txt.
Raw Trace:

```
find . -maxdepth 1 -type f -printf '%P\n' > file_list.txt
(OS response: empty)
```

Step 2
Summary: Create the matches directory.
Raw Trace:

```
mkdir -p matches
(OS response: empty)
```

Step 3 (FAILURE STEP)
Summary: Incorrectly concludes that no files exist due to empty console output.
Raw Trace:

```
answer(0)
```

Planning Error

Step 1
Summary: Compute the sum of integers from validated_integers.txt.
Raw Trace:

```
awk '{s+=$0} END{print s+0}' validated_integers.txt
-> 0
```

Step 2
Summary: Inspect validated_integers.txt contents.
Raw Trace:

```
cat -n validated_integers.txt
```

Step 3 (FAILURE STEP)
Summary: Runs an unrelated command counting files under /etc.
Raw Trace:

```
find /etc -maxdepth 1 -type f | wc -l
-> 54
```

Step 4
Summary: Reports the unrelated result as the final answer.
Raw Trace:

```
answer(54)
```

Memory Limitation

Step 1
Summary: On the My Orders page, the agent reads the latest cancelled order and correctly identifies the total cost as $365.42.
Raw Trace:

```
[Step 1]
Page : My Orders

Agent: The latest cancelled order is order 000000170
dated 5/17/23 with total $365.42.

Action: click [45873]  # Video Games
```

Step 2
Summary: After navigating to the Video Games page, the agent behaves as if it does not know the cancelled order total and decides to open the account page to retrieve it again.
Raw Trace:

```
[Step 2]
Page : Video Games

Agent: To find the latest cancelled order, I need to open
the account page where the order history is located.

Action: click [48620]  # My Account
```

Step 3
Summary: The agent opens the order history again to search for the cancelled order, even though the total was already observed earlier.
Raw Trace:

```
[Step 3]
Page : My Account

Agent: I should first open the order history to find the
latest cancelled order.

Action: click [51825]  # My Orders
```

Step 4 (FAILURE STEP)
Summary: Instead of using the already known value ($365.42), the agent continues navigating away from the orders page and repeatedly attempts to retrieve the same information, indicating that the earlier observation is no longer retained in its working memory.
Raw Trace:

```
[Step 4]
Page : My Orders

Agent: I will browse the Video Games category and then
report the total of the latest cancelled order.

Action: click [53661]  # Video Games
```

E.3 Embodied

Instruction

Step 1
Summary: Initialize by homing the arm.
Raw Trace:

```
home({})
```

Step 2 (FAILURE STEP)
Summary: Moves to a fixed pose assuming the target is the red cube without verifying the object color.
Raw Trace:

```
move_to_pose({'arm': 'right',
'pose': [0.2, 0.06, 1.0, 0, -0.818, 0.574, 0]})
```

Step 3
Summary: Attempts to grasp at the assumed target location.
Raw Trace:

```
envelop_grasp({'arm': 'right'})
```

Step 4
Summary: Lifts the object after the grasp without confirming object identity.
Raw Trace:

```
move_to_pose({'arm': 'right',
'pose': [0.2, 0.06, 1.1, 0, -0.818, 0.574, 0]})
```

False Assumption

Step 1
Summary: Return the arm to the home position.
Raw Trace:

```
home({})
```

Step 2 (FAILURE STEP)
Summary: Moves to a hardcoded pose assuming it corresponds to the blue cube without verifying the object.
Raw Trace:

```
move_to_pose({'arm': 'right',
'pose': [0.3, 0.26, 1.0, 0, -0.818, 0.574, 0]})
```

Step 3
Summary: Attempts to grasp whatever object is at the assumed cube location.
Raw Trace:

```
envelop_grasp({'arm': 'right'})
```

Step 4
Summary: Lifts the grasped object after the unverified pick.
Raw Trace:

```
move_to_pose({'arm': 'right',
'pose': [0.3, 0.26, 1.1, 0, -0.818, 0.574, 0]})
```

Planning Error

Step 1
Summary: Moves the right arm toward the first cube location.
Raw Trace:

```
move_to_pose({'arm': 'right',
'pose': [0.2, 0.06, 1.0, 0, -0.818, 0.574, 0]})
```

Step 2
Summary: Grasps the first cube with the right arm.
Raw Trace:

```
envelop_grasp({'arm': 'right'})
```

Step 3 (FAILURE STEP)
Summary: Attempts to move to the second cube while still holding the first cube.
Raw Trace:

```
move_to_pose({'arm': 'right',
'pose': [0.1, 0.16, 1.0, 0, -0.818, 0.574, 0]})
```

Step 4
Summary: Attempts a second grasp while the gripper is already occupied.
Raw Trace:

```
envelop_grasp({'arm': 'right'})
```

Step 5
Summary: Moves toward another cube again without releasing or placing the previous one.
Raw Trace:

```
move_to_pose({'arm': 'right',
'pose': [0.3, 0.26, 1.0, 0, -0.818, 0.574, 0]})
```

E.4 Database

Instruction

Step 1
Summary: The question requests account numbers of the oldest female clients with the lowest average salary.
Raw Trace:

```
Question:
Name the account numbers of female clients who are oldest
and have lowest average salary?
```

Step 2
Summary: The evidence clarifies schema meanings for gender and salary.
Raw Trace:

```
Evidence:
Female refers to 'F' in the gender;
A11 contains information about average salary.
```

Step 3 (FAILURE STEP)
Summary: The agent SQL filters only the global minimum salary district and then selects the oldest female within that subset.
Raw Trace:

```
Agent SQL:
SELECT A.account_id
FROM client C
JOIN disp D ON C.client_id = D.client_id
JOIN account A ON D.account_id = A.account_id
JOIN district R ON C.district_id = R.district_id
WHERE C.gender = 'F'
AND R.A11 = (SELECT MIN(A11) FROM district)
AND C.birth_date =
(
 SELECT MIN(C2.birth_date)
 FROM client C2
 JOIN district R2 ON C2.district_id = R2.district_id
 WHERE C2.gender = 'F'
 AND R2.A11 = (SELECT MIN(A11) FROM district)
);
```

Step 4
Summary: Gold SQL orders by birth_date (oldest first) then salary and limits to one result.
Raw Trace:

```
Gold SQL:
SELECT T3.account_id
FROM client T1
JOIN district T2 ON T1.district_id = T2.district_id
```

```
JOIN account T3 ON T2.district_id = T3.district_id
JOIN disp T4 ON T1.client_id = T4.client_id
AND T4.account_id = T3.account_id
WHERE T1.gender = 'F'
ORDER BY T1.birth_date ASC, T2.A11 ASC
LIMIT 1;
```

Step 5
Summary: The system marks the agent's answer as incorrect.
Raw Trace:

```
Correct : False
```

False Assumption

Step 1
Summary: User asks for statistics about female clients in qualifying districts with salary and loan constraints.
Raw Trace:

```
Question:
What are the overall statistics for female clients in districts
where the average salary is between 6,000 and 10,000,
ranking in the top 3 for salary within their region,
having at least 5 female clients, and with active loan accounts?
```

Step 2
Summary: Evidence clarifies schema meanings for salary, gender, and loan status.
Raw Trace:

```
Evidence:
Average salary refers to A11
Female refers to gender = 'F'
Loan status:
'A' = active
'B' = completed
'C' = defaulted
```

Step 3 (FAILURE STEP)
Summary: The agent constructs SQL that incorrectly joins account on **district_id** and computes unrelated birth-date statistics.
Raw Trace:

```
Agent SQL:
SELECT COUNT(*) AS female_count,
MIN(birth_date) AS oldest_birth_date,
MAX(birth_date) AS youngest_birth_date
FROM client
WHERE gender = 'F'
AND district_id IN ( ... )
```

Step 4
Summary: Gold SQL correctly computes district-level statistics using CTEs and joins.
Raw Trace:

```
Gold SQL:
WITH DistrictStats AS (...)
SELECT COUNT(DISTINCT ds.district_id),
SUM(ds.female_clients),
AVG(ds.avg_salary),
...
FROM DistrictStats ds
JOIN AccountActivity aa
ON ds.district_id = aa.district_id;
```

Step 5
Summary: The system indicates the answer is incorrect.
Raw Trace:

```
Correct : False
```

Planning Error

Step 1
Summary: User asks for multiple statistics about high schools in Amador County.
Raw Trace:

```
Question:
What are the key statistics for high schools in Amador County,
including enrollment, charter breakdown, SAT metrics,
largest school, and high-poverty counts?
```

Step 2
Summary: Definitions clarify high school grades, poverty threshold, and charter flag.
Raw Trace:

```
Evidence:
High schools: Low Grade = '9' AND High Grade = '12'
High poverty: Percent Eligible FRPM > 0.75
Charter schools: Charter School = 1
```

Step 3 (FAILURE STEP)
Summary: Agent produces only a single query counting high-poverty schools.
Raw Trace:

```
Agent SQL:
SELECT COUNT(*)
FROM frpm
WHERE County Name = 'Amador'
```

```
AND Low Grade = '9'
AND High Grade = '12'
AND Percent Eligible FRPM > 0.75;
```

Step 4
Summary: Gold SQL computes all required metrics using multiple CTEs.
Raw Trace:

```
Gold SQL:
WITH SchoolInfo AS (...),
SATData AS (...)
SELECT COUNT(...), AVG(...), ...
FROM SchoolInfo;
```

Step 5
Summary: Notes indicate the model failed to decompose the multi-part request.
Raw Trace:

```
Notes:
Model failed to decompose the request and produced
only a single simple query.
```

Planning Error

Step 1
Summary: User asks for multiple statistics about high schools in Amador County.
Raw Trace:

```
Question:
What are the key statistics for high schools in Amador County,
including enrollment, charter breakdown, SAT metrics,
largest school, and high-poverty counts?
```

Step 2
Summary: Definitions clarify high school grades, poverty threshold, and charter flag.
Raw Trace:

```
Evidence:
High schools: Low Grade = '9' AND High Grade = '12'
High poverty: Percent Eligible FRPM > 0.75
Charter schools: Charter School = 1
```

Step 3 (FAILURE STEP)
Summary: Agent produces only a single query counting high-poverty schools.
Raw Trace:

```
Agent SQL:
SELECT COUNT(*)
FROM frpm
WHERE County Name = 'Amador'
AND Low Grade = '9'
AND High Grade = '12'
AND Percent Eligible FRPM > 0.75;
```

Step 4
Summary: Gold SQL computes all required metrics using multiple CTEs.
Raw Trace:

```
Gold SQL:
WITH SchoolInfo AS (...),
SATData AS (...)
SELECT COUNT(...), AVG(...), ...
FROM SchoolInfo;
```

Step 5
Summary: Notes indicate the model failed to decompose the multi-part request.
Raw Trace:

```
Notes:
Model failed to decompose the request and produced
only a single simple query.
```

E.5 How We Construct and Label Failure Mini-Trajectories

Source and search procedure. Our examples are WebArena-style failure turning points expressed in the canonical web-agent interaction schema (goal + observation/accessibility-tree snippet + action), and organized by WebArena's five domains (Shopping, Shopping Admin/CMS, GitLab, Reddit, Map). We first seeded candidate failure patterns by reviewing public analyses of web-agent traces that report recurring breakdown modes such as ineffective UI interactions (no state change loops), input accumulation due to non-overwriting typing semantics, and hallucinated form fields. We then *instantiated* each seeded pattern into a domain-specific micro-scenario by (i) choosing a domain-consistent subgoal (e.g., sorting, filtering, form submission, route planning), (ii) writing a minimal observation snippet that contains the critical UI affordances and state signals needed to trigger the failure, and (iii) emitting a 2–4 step action sequence that reaches the failure point (the first irreversible divergence or loop onset). This yields a compact catalog of representative turning points while preserving the execution semantics and action space assumed by common WebArena baselines.

What counts as long-horizon. We define a task as *long-horizon* if correct completion requires either (a) **multi-step dependency** where earlier constraints must be retained and enforced after intervening navigation/interaction, or (b) **multi-subgoal composition** (e.g., discover → verify → write/-submit) that cannot be solved by a single local decision. Operationally, our appendix marks a failure as long-horizon when the erroneous action depends on (i) an earlier constraint or intermediate result that is no longer in the immediate observation window, or (ii) a missing prerequisite subgoal that should have been executed earlier (e.g., sorting before selecting the "top" comment).

Failure type assignment. We label each mini-trajectory with one of seven failure types using a lightweight decision rule that prioritizes *proximal cause* at the turning point: (1) **Environment disturbance** when the UI/state changes render the agent's action ineffective or stale (e.g., modal overlays, session expiry, non-responsive dropdowns); (2) **Instruction misunderstanding** when the action violates an explicit user constraint ("do not purchase", "walking not driving"); (3) **Catastrophic forgetting** when an early constraint is dropped after intervening steps; (4) **False assumptions** when missing information is fabricated (e.g., user identity fields, absent emails); (5) **Planning errors** when prerequisite subgoals are skipped or ordered incorrectly; (6) **History error accumulation** when small state-tracking errors compound (e.g., append-typing, toggle oscillation); and (7) **Memory limitations** when the agent cannot retain or aggregate required multi-item evidence across steps. These categories are consistent with diagnostic perspectives that seek to attribute failures to specific breakdown loci in interactive web settings.

Why we only show 2 - 4 steps. Although the underlying tasks are long-horizon, the *failure turning point* is typically localized: a brief window where the agent first enters an unrecoverable branch (constraint violation) or an absorbing loop (no-state-change / oscillation), which makes a 2 - 4 step slice an adequate witness of the long-horizon breakdown.

```
if (action has no effect OR UI state
    ↪ changed/stale
    ↪ element/overlay/session):
    ↪ ENV_DIST
else if (action violates explicit
    ↪ constraint OR wrong
    ↪ modality/target): INSTR_MIS
else if (fabricates missing fields OR
    ↪ claims unseen facts):
    ↪ FALSE_ASSUME
else if (drops earlier constraint /
    ↪ missing prerequisite /
    ↪ compounding drift):
    ↪ FORGET/PLAN/HIST/MEM by nearest
    ↪ cause
```

F Appendix: Experiment 2 (LLM-as-a-Judge)

F.1 Attribution Procedure

At the per-step level, annotators assign a single label via the decision tree, which organizes Q1–Q7 into four conceptual stages following a causal hierarchy from external inputs to internal behavior: (1) *External Factors* (Q1–Q2; environment, instruction) → (2) *Information Availability* (Q3–Q4; what the agent retains in its working context) → (3) *Reasoning* (Q5–Q6; how the agent uses that information) → (4) *Propagation* (Q7; cross-step accumulation). For each failure step in temporal order, annotators walk Q1–Q7 sequentially and assign the label of the first "Yes" branch. This top-down ordering resolves common category overlaps by giving upstream causes precedence over downstream symptoms. We additionally include an †*Others* bucket as a safety valve for failures whose primary cause does not fall into any of the seven categories.

F.2 Why human validation is restricted to 40 trajectories per study

Human annotation at full scale is infeasible due to three compounding factors.

Trajectory length and density. A single Web trajectory has a median length of ∼6300 words (≈25 pages) of interleaved DOM observations and agent actions; OS trajectories average ∼1800 words of shell I/O whose interpretation requires reconstructing the system state at each step. At an optimistic 5 minutes per trajectory, annotating the full 3100+ trajectory pool would require hundreds of person-hours per annotator.

Taxonomy complexity. Each trajectory must be mapped to one of 7 top-level categories with 11 subcategories, each defined by a multi-sentence specification plus long-horizon rationale. The annotator must both localize the failure step and match its semantics against fine-grained definitions—a labeling decision that itself takes several minutes of re-reading.

Domain expertise. Database annotations require reading SQL with CTEs and multi-table joins to distinguish schema hallucination from ambiguous instruction; embodied annotations require evaluating primitive-action plans against simulator affordances. No single annotator is fluent across all four domains, so per-domain experts are needed, further multiplying recruitment cost.

We therefore restrict each validation study to a stratified 40-trajectory sample, balancing across domains and failure categories to preserve representativeness.

F.3 On the gap between human–judge ($\kappa = 0.84$) and human–human ($\kappa = 0.61$) agreement

The two (κ) values measure different quantities and should not be interpreted as bounding each other. Both are reported at the same 7-category granularity to ensure comparability.

Different sources of variance. H–H ($\kappa = 0.61$) captures inter-human interpretive variance: two independent experts navigating edge cases of a 7-way long-horizon failure taxonomy with their own interpretive priors. Under standard interpretation of (κ), 0.61 corresponds to substantial agreement and is consistent with values reported for comparably fine-grained agent-trajectory taxonomies. H–J ($\kappa = 0.84$) instead measures whether the judge faithfully implements the structured rubric against a rubric-calibrated annotator. The judge applies the rubric uniformly, whereas two humans introduce independent interpretive noise; consequently the judge can align with a single rubric-trained annotator more closely than two humans align with each other.

Interpretive ceiling. The intuition that ($\kappa_{H-J} \leq \kappa_{H-H}$) holds only when humans constitute the gold standard. In our setup the gold standard is the written taxonomy rubric; humans approach but do not perfectly realize it. A rubric-consistent judge can therefore exceed inter-human agreement without implying that the judge is "more accurate" than humans—only that it is more rubric-consistent than two humans are with each other.

Different samples and annotators. H–H agreement is computed between two expert annotators on a 40-trajectory sample drawn to stress-test taxonomy interpretability. H–J agreement is computed between the LLM-as-a-Judge and a third co-author who labeled a separate 40-trajectory sample independently of the first two annotators. The two studies share neither the sample nor the annotators, so the two (κ) values quantify variance in two distinct comparisons rather than a single shared ceiling.

F.4 Results

We apply the trajectory-grounded LLM-as-a-Judge pipeline to the full set of 3100+ trajectories spanning all four domains and both model families. Figures 9, 10, and 11 report the resulting failure-mode distributions broken down by domain, by model, and by domain × model, respectively.

Two qualitative patterns are worth highlighting. First, failures are far from uniformly distributed across the seven categories: *Planning Error*, *Memory Limitation*, and *Catastrophic Forgetting* together account for the majority of attributed failures in every domain, while the remaining categories take up a noticeably smaller share. Second, the two model families exhibit qualitatively different failure profiles: GPT failures are dominated by Planning Error and Memory Limitation, whereas Claude shows a relatively larger share of Environment and Instruction failures, with Planning Error still the single most common category.

F.5 Scope and Complementary Dimensions

HORIZON characterizes what fails at each step and is therefore complementary to—not a replacement for—evaluation dimensions that target other facets of long-horizon competence: recovery and self-correction (e.g., correction latency or recovery rate after a detected error), action efficiency (e.g., actual steps relative to the optimal horizon H^*), failure cascade depth (the length of compound failure chains such as False Assumption → Planning Error → History Error Accumulation), and grounding / verification frequency (how often an agent re-checks its assumptions or re-observes the environment). We view these as natural extensions of the present taxonomy and leave their integration to future work.

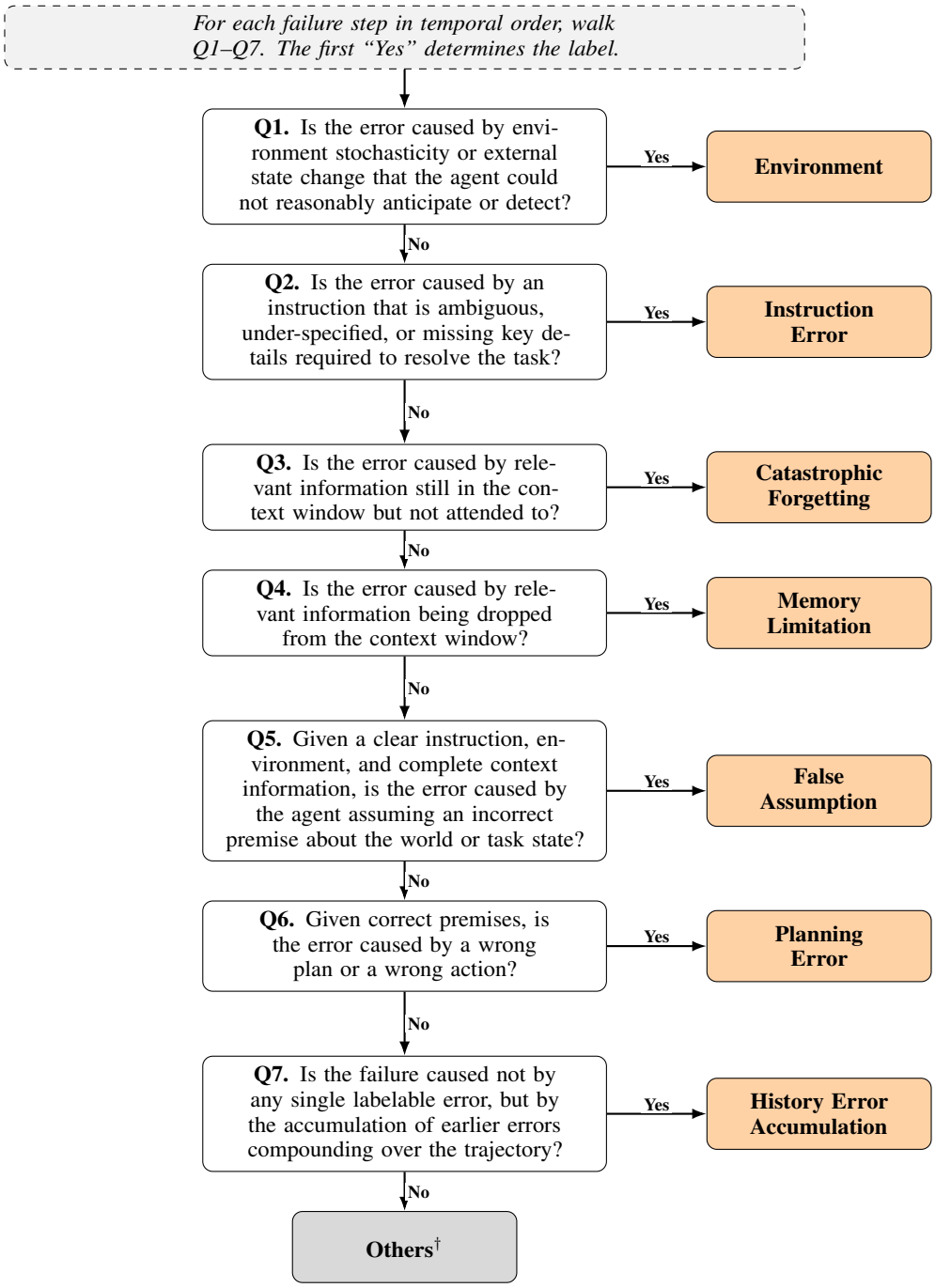

Figure 8: Decision tree for failure attribution. The seven categories are organized into four conceptual stages following a causal hierarchy from external inputs to internal behavior: **External Factors** (Q1–Q2; environment, instruction) → **Information Availability** (Q3–Q4; what the agent retains in its working context) → **Reasoning** (Q5–Q6; how the agent uses that information) → **Propagation** (Q7; cross-step accumulation). For each failure step in temporal order, annotators walk Q1–Q7 sequentially; the first "Yes" branch determines the label. This top-down ordering resolves common category overlaps by giving upstream causes precedence over downstream symptoms—e.g., a wrong inference is attributed to memory loss (Q3/Q4) rather than False Assumption (Q5) when the relevant information is no longer attended to or has been dropped from context. †*Others* is both a safety-valve bucket for failures outside the seven categories and an explicit marker of taxonomy scope: we attribute *what* failed at each step, orthogonal to complementary evaluation dimensions (recovery / self-correction, action efficiency, failure cascade depth, grounding / verification frequency) which we leave to future work.

Figure 9: Failure mode distribution across the four task domains

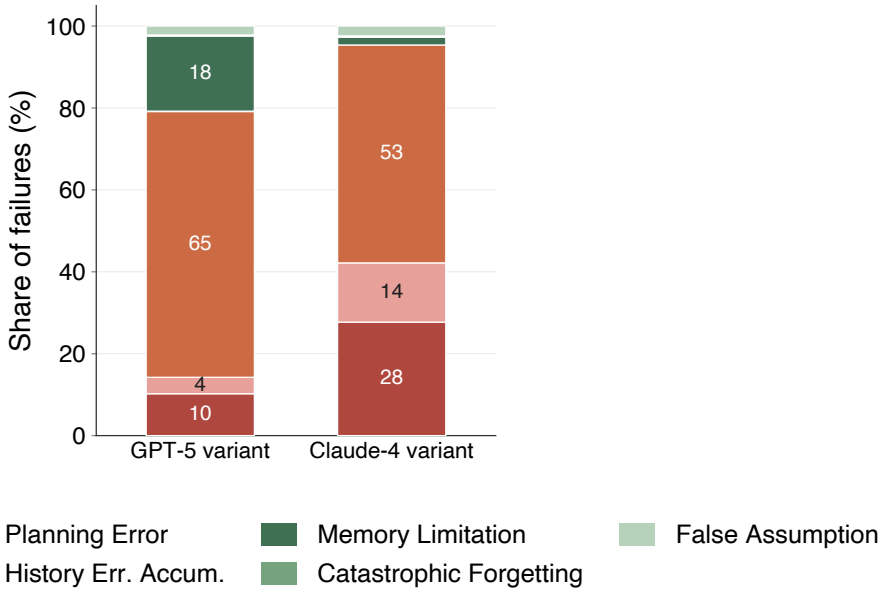

Figure 10: Failure mode distribution for GPT-5-mini vs. Claude-4-Sonnet aggregated across all domains.

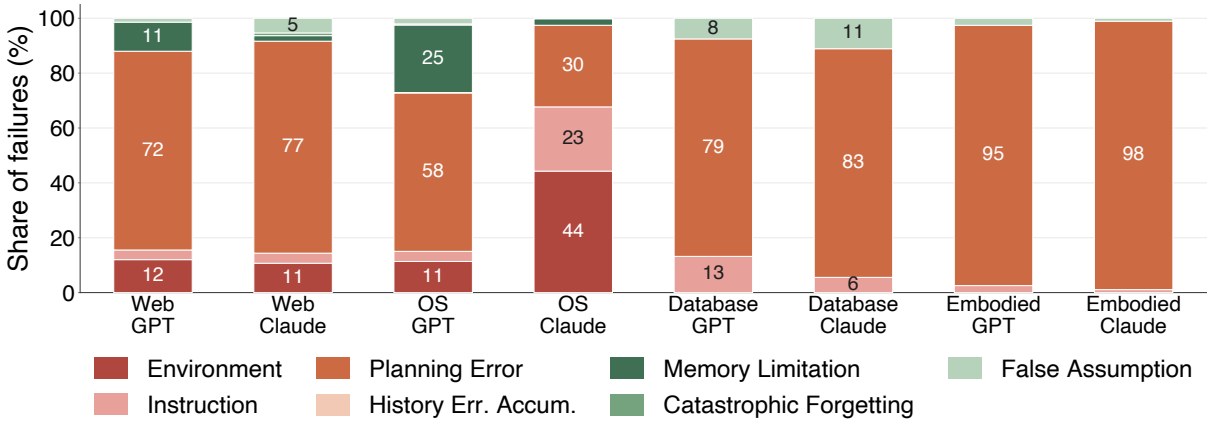

Figure 11: Failure mode distribution broken down by domain and model.

F.6 Prompt

We attribute failures using a single combined system prompt that asks the LLM judge to (i) walk the Q1–Q7 decision tree of Figure 8 and assign one of the seven categories (Stage 1), and (ii) extract a contextual *failure slice* grounded in the raw trajectory (Stage 2). The full prompt is reproduced verbatim below. The placeholder `{FAILURE_TYPES_TEXT}` is filled at runtime with the per-category definitions in Appendix C; unicode separators and arrows in the original have been transliterated to ASCII for typesetting.

Judge System Prompt (combined Stage 1 + Stage 2)

```
You are an expert evaluator for long-horizon LLM agents.
Your task has two parts -- complete BOTH in a single response.

==============================
STAGE 1 -- Failure Classification (Q1-Q7 Decision Tree)
==============================

If the trajectory is SUCCESSFUL, use "failure_type": "Success" and stop.

Otherwise: walk through the FAILURE STEPS in temporal order. For the
EARLIEST critical failure step, apply the questions Q1-Q7 below IN ORDER.
The FIRST question answered "Yes" determines the label. Do not skip or
reorder questions. If none are "Yes", label as "Others".

DETAILED DEFINITIONS (for reference when judging each question):
{FAILURE_TYPES_TEXT}

DECISION TREE -- ask each question in order, stop at first "Yes":

  Q1. Was the failure caused by an environment disturbance NOT caused by
      the agent's own action? (e.g., flaky tool/API, missing file the
      agent did not delete, perception noise, page changed without agent
      action)
        -> Yes => "Environment"

  Q2. Is the instruction itself ambiguous, under-specified, or missing
      key semantics such that even a perfect agent could not infer the
      correct goal?
        -> Yes => "Instruction"

  Q3. Does the agent hold a WRONG PREMISE about the world or task state
      at this step (a factual belief that is false, e.g., assuming a
      column exists, assuming an object is in a location, hallucinating
      a tool result)?
        -> Yes => "False Assumption"

  Q4. Given a CORRECT premise, did the agent derive a wrong plan /
      action choice (logic, ordering, missing sub-goal, invalid action
      selection)?
        -> Yes => "Planning Error"

  Q5. Is the relevant information STILL PRESENT in the agent's context
      window at this step, but the agent failed to attend to it (it was
      not truncated, the agent just ignored or contradicted recently-
      visible info)?
        -> Yes => "Catastrophic Forgetting"

  Q6. Was the relevant information DROPPED from the context window due
      to token-budget limits (truncation, summarization, or long-horizon
      length), so the agent could not see it at this step?
        -> Yes => "Memory Limitation"

  Q7. Does this step introduce NO NEW error of its own -- it only
      propagates an error that was already attributed to an earlier
      step?
        -> Yes => "History Error Accumulation"

  None of the above => "Others"

Rules:
```

```
- failure_type must be EXACTLY one of: the 7 categories above,
  "Success", or "Others".
- Focus on ROOT CAUSE at the earliest critical failure step -- do not
  relabel the trajectory by symptoms of later steps.
- Do NOT have a prior toward any single category. Q1-Q7 must be
  considered independently and in order; only assign Memory Limitation
  when Q6 is clearly true AND Q1-Q5 are clearly not.
- Distinguishing Q5 vs Q6: if the dropped/forgotten information was
  visible within roughly the last few steps of context (still inside
  the window), it is Q5 (Catastrophic Forgetting); only if it was
  clearly outside the active window due to length/truncation is it Q6
  (Memory Limitation).
- For embodied/robot plans: evaluate whether the primitive action
  sequence correctly achieves the task; Q3 covers wrong world-state
  premises, Q4 covers wrong plan derivation given correct premises.

CONTEXT FOR Q5 / Q6:
You are reading the FULL recorded trajectory. The agent itself operated
under a LIMITED context window -- earlier steps may have been truncated
or summarized away. When deciding between Q5 and Q6, judge whether the
missing info plausibly fits in a typical context window at the failure
step, not whether YOU can see it in the full record.

==============================
STAGE 2 -- Extract Failure Context Slice
==============================

1. Identify the SPECIFIC STEP where the critical failure first occurs.

2. DEFAULT -- Extract a window of up to 5 steps:
   - 2 steps BEFORE the failure step (if available)
   - The failure step itself
   - 2 steps AFTER the failure step (if available)
   If fewer than 2 steps exist before/after, include as many as
   available.

   SPECIAL RULES for certain failure types -- extract MORE steps to
   capture the full pattern:

   If failure_type == "Memory Limitation":
   - This failure spans the ENTIRE trajectory -- you MUST include at
     least 3 steps:
     (A) ESTABLISHMENT step: the EARLIEST step where the key information
         was first stated (e.g., the step that mentions the specific
         value, constraint, or observation that is later lost)
     (B) DIVERGENCE step: a step where there is evidence the context was
         truncated or summarized (e.g., the agent re-asks for something,
         contradicts itself, or a summary omits details)
     (C) FAILURE step: the step where the agent acts incorrectly due to
         missing memory
   - Label them clearly: "establishment step", "divergence step",
     "failure step"
   - Add additional surrounding context steps if needed (up to 8 steps
     total)
   - failure_step_number should point to the FAILURE step (C)

   If failure_type == "History Error Accumulation":
   - You MUST show the CHAIN of errors, not just the final step:
     (A) ORIGIN step: the first mistake or incorrect action that seeds
         the error
     (B) ACCUMULATION steps: 2-3 intermediate steps showing the error
         compounding
     (C) FAILURE step: the final step where the accumulated errors cause
         task failure
   - Label them: "origin step", "accumulation step N", "failure step"
   - Include up to 8 steps total to capture the full progression
   - failure_step_number should point to the ORIGIN step (A)

3. For EACH step output BOTH:
```

```
   (a) A ONE-SENTENCE SUMMARY -- concise abstraction
   (b) The ORIGINAL RAW TEXT -- exact verbatim copy from the trajectory

Rules:
- The "raw" field MUST be EXACT verbatim text -- do NOT paraphrase.
- Handle each step INDEPENDENTLY -- do not combine steps.
- For SUCCESS: set failure_step_number to null and steps to [].
- CRITICAL: For "Memory Limitation" and "History Error Accumulation",
  steps MUST NOT be empty. If you cannot find specific steps, still
  include the most relevant steps from the trajectory.

==============================
OUTPUT FORMAT -- strict JSON, nothing else:
==============================
{
  "failure_type": "<one of the 7 categories above, 'Others', or
                    'Success'>",
  "reason": "<one-sentence explanation grounded in the trajectory;
             cite which Q (Q1..Q7) was the first Yes>",
  "failure_step_number": <integer, 1-indexed position in full
                          trajectory, or null>,
  "steps": [
    {"label": "step 1",
     "summary": "<one sentence>", "raw": "<exact text>"},
    {"label": "step 2",
     "summary": "<one sentence>", "raw": "<exact text>"},
    {"label": "step 3 (FAILURE STEP)",
     "summary": "<one sentence>", "raw": "<exact text>"},
    {"label": "step 4",
     "summary": "<one sentence>", "raw": "<exact text>"},
    {"label": "step 5",
     "summary": "<one sentence>", "raw": "<exact text>"}
  ]
}

Output ONLY valid JSON with exactly these four keys.
```

G Appendix: More Discussion

This work aims to advance nuanced discussions of systematic long-horizon analysis and to provide methodological guidance for scalable, reproducible, and cross-domain research on long-horizon agentic AI. Below, we discuss the implications of findings in Section 4 and outline directions for future research.

Empirical Support for HORIZON. First, we observe a consistent pattern of performance collapse beyond early horizon extension levels across all four domains, providing mechanistic evidence that increasing s indeed pushes agents into regimes dominated by long-horizon reasoning failure. This sharp transition indicates that our horizontal horizon construction meaningfully orders tasks by long-horizon difficulty. In parallel, qualitative analysis of failed trajectories shows that failures can be systematically categorized using the seven-category taxonomy defined in HORIZON (Table 3), providing phenomenological structure to observed breakdowns. Notably, all annotated failures observed across domains in our experiments fall within this taxonomy, despite substantial differences in task modality and environment dynamics. This suggests that HORIZON provides a valid and generalizable framework for studying long-horizon tasks across domains.

Long-Horizon Tasks Lack a Canonical Definition. A central challenge highlighted by our study is that long-horizon tasks remain poorly defined in prior literature. Existing benchmarks operationalize "horizon" using incompatible proxies, such as interaction length, number of tool calls, reasoning steps, or episode duration, often without explicit justification. As a result, what constitutes a long-horizon task in one domain may be trivial, incomparable, or ill-defined in another. Our findings reinforce that horizon cannot be normalized by step count alone and must instead be grounded in task structure. This motivates the need for principled, task-centric definitions, as adopted in HORIZON, to enable meaningful cross-domain comparison.

Long-Horizon Failure Reflects Fundamental Generalization Limits. Long-horizon failures should not be viewed merely as extensions of short-horizon planning errors. Instead, they reflect a distinct form of generalization under distribution shift, where agents must operate under compounding changes in state, action consequences, and goal dependencies. As horizons increase, agents encounter states and decision contexts that diverge increasingly from those seen earlier in the trajectory. We suggest that breakdowns in long-horizon settings arise not only from insufficient planning depth, but from systematic limits in maintaining and updating beliefs over extended interactions. This reframes long-horizon reasoning as a core generalization challenge rather than a purely algorithmic planning problem.

Why Scaling Alone Is Unlikely to Solve Long-Horizon Failures. Although frontier models outperform smaller systems at moderate horizons, our results show that performance gaps often collapse once agents enter the long-horizon failure regime. This convergence suggests diminishing returns from additional model capacity alone. Importantly, different failure types demand qualitatively different interventions: environment disturbances require monitoring and recovery mechanisms, catastrophic forgetting calls for improved memory and constraint tracking, while planning errors may benefit from better sub-goal decomposition. Therefore, one-size-fits-all solution, such as increasing model size or training on longer traces, is unlikely to address the root causes of long-horizon failure. Systematic diagnosis is a prerequisite for targeted improvement.

Breaking Points as Transition Regions, Not Thresholds. Our empirical findings support viewing breaking points as transition regions rather than precise thresholds. Across domains, performance collapse occurs abruptly but at different horizons, and often spans a narrow range of extension levels rather than a single step. This reconciles the intuitive appeal of breaking points with our theoretical argument against universal definitions. Operationally, it suggests that evaluation should focus on identifying regions where failure dynamics qualitatively change, rather than attempting to pinpoint exact breaking steps. Such region-based analysis is more robust to domain variation and model heterogeneity.

Implications for Benchmark Design and Agent Evaluation. Taken together, our results call for a shift in how long-horizon agents are evaluated. Reporting single-point accuracy obscures where and how failures emerge as horizons grow. Horizon-aware evaluation—reporting performance curves, failure transitions, and attributed failure types—provides a richer and more actionable picture of agent behavior. By decoupling horizon measurement from failure attribution, HORIZON offers a flexible framework for studying long-horizon reasoning without assuming a universal breaking point. We hope this perspective encourages the community to design benchmarks and diagnostics that better reflect the realities of long-horizon agent deployment.