

DYNSTEER: DYNAMIC STAGE-WISE TRAJECTORY EVALUATION AND EXECUTION-TIME REVIEW FOR AGENTS

Zhichao Shi^{1,2,3,4}, Xuhui Jiang^{3,4*}, Wenjie Zhang⁵, Xiaojun Wu^{3,4,6}, Cehao Yang^{3,4,6},
Chengjin Xu^{3,4}, Jian Guo⁴, Yuanzhuo Wang^{2*}

¹ School of Advanced Interdisciplinary Sciences, UCAS

² State Key Lab of AI Safety, Institute of Computing Technology, CAS

³ DataArc Tech Ltd.

⁴ IDEA Research, International Digital Economy Academy

⁵ Jiangnan University

⁶ The Hong Kong University of Science and Technology (Guangzhou)

ABSTRACT

Large language model agents are increasingly deployed for long-horizon task execution, raising a central granularity question for trajectory evaluation: whole-trajectory verification is too coarse to capture concrete failures and their associated evidence in long trajectories, while atomic-step scoring is too fine-grained, noise-sensitive, and computationally expensive. This granularity gap makes a single-reference trajectory paradigm inadequate for assessing the rich space of valid agent execution paths and delays timely feedback and early stopping in long-horizon tasks. To address these issues, we propose DynSTEER, a dynamic stage-wise framework for agent trajectory evaluation. DynSTEER bridges the granularity gap through stage-wise dynamic evaluation that segments rollouts at key execution nodes and adapts its multi-tier review strategy based on stage-level results; it compiles a path-tolerant milestone graph from public task views to preserve diverse legal paths without reference leakage; and it supports terminating unrecoverable agent executions to curb resource waste. Experiments show that DynSTEER improves evaluation discriminability by 85.2% over native evaluation, separates all model pairs with statistical significance, and saves 45.41% of execution steps on failed rollouts.

1 INTRODUCTION

LLM agents are increasingly undertaking complex, long-horizon tasks across real-world environments (Yao et al., 2023; Shinn et al., 2023; Jimenez et al., 2024; Gu et al., 2024; Zhou et al., 2024). As task horizons expand, evaluating only final output results becomes insufficient to identify agent deficiencies and diagnose failure modes, therefore bringing trajectory-level evaluation to the forefront of agent research (Qian et al., 2024; Shi et al., 2026).

However, existing trajectory evaluation paradigms face a central granularity problem, especially for long-horizon tasks. Whole-trajectory verification is too coarse to focus on concrete failures and their associated evidence in long trajectories, whereas atomic-step scoring is too fine-grained: isolated messages lack task context, so ordinary retries and harmless observations become noise. This granularity gap further raises two challenges. Agent executions often admit multiple valid and effective paths. Because of this granularity gap, evaluation typically offers only limited path references and therefore struggles to faithfully reflect the quality of diverse agent executions. Moreover, full-trajectory review delays diagnosis that could be made earlier, so an execution that has already gone wrong may continue, wasting model resources and triggering additional skill or tool calls.

As illustrated in Figure 1b, we introduce **DynSTEER**, a **DYNAMIC Stage-wise Trajectory Evaluation and Execution-time Review** framework. To address the granularity challenge, DynSTEER performs

*Corresponding author

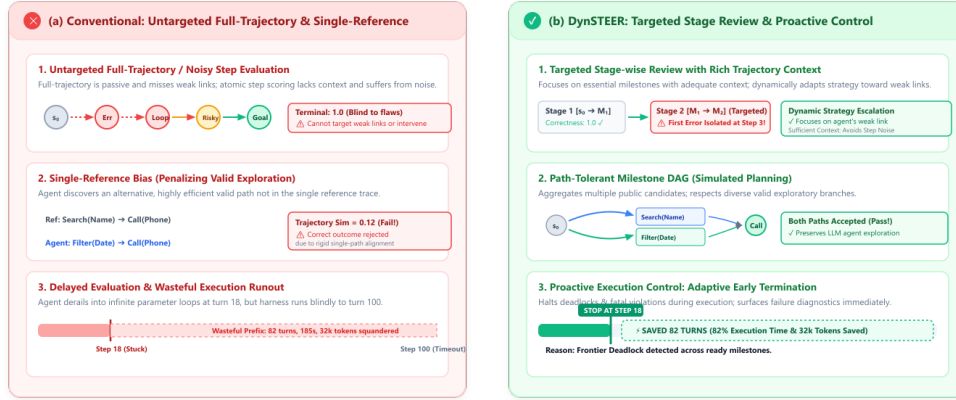


Figure 1: **Comparison of Agent Evaluation Paradigms.** (a) Conventional paradigms evaluate either coarse outcomes or noisy atomic steps, often against a single reference, and discover waste only after execution. (b) DYNSTEER evaluates path-tolerant milestone-bounded stages and uses the resulting stage evidence to target review and stop hopeless prefixes early.

stage-wise dynamic evaluation: it segments rollouts at necessary execution nodes, preserves task context while isolating localized evidence for error attribution, and adapts its multi-tier review strategy based on stage-level results. To avoid single-reference bias, it compiles a path-tolerant milestone graph from public task views via multi-candidate simulated planning and majority consensus, capturing necessary task invariants while accommodating diverse legitimate paths. Finally, it supports detecting unrecoverable prefixes and terminating them to curb resource waste.

Experiments demonstrate that DynSTEER improves evaluation discriminability by 85.2% over native evaluation, enhances diagnostic precision, separates all model pairs with statistical significance, preserves the validity of diverse execution paths, and saves 45.41% of execution steps on failed rollouts.

In summary, our contributions are as follows:

- **Stage-wise Dynamic Evaluation:** We segment rollouts at necessary execution-step nodes and adapt multi-tier review routing from stage-level results, isolating context-rich evidence windows for multidimensional scoring and first-error localization.
- **Path-Tolerant Task Blueprint Compilation:** We compile a milestone DAG from public task views via multi-candidate planning and strict-majority consensus, preserving diverse valid paths without reference leakage.
- **Counterfactual Early Termination:** We detect unrecoverable prefixes from stage-level evidence and terminate them, substantially reducing wasted computation.

2 RELATED WORK

2.1 AGENT TRAJECTORY EVALUATION

Unlike outcome benchmarks that inspect only terminal code patches or database records (Jimenez et al., 2024; Qin et al., 2024; Liu et al., 2024; Zhou et al., 2024), trajectory evaluation examines intermediate planning decisions, tool invocations, and error recoveries. Recent benchmarks expand trajectory assessment: BFCL (Patil et al., 2025) evaluates multi-step tool calls via AST checks; TRAJECT-Bench (Li et al., 2025) assesses tool selection and dependency order; AgentRewardBench (Zhou et al., 2025) benchmarks trajectory judges; and DeepRed (Kuo et al., 2026) and Claw-Eval (Zhang et al., 2026) leverage environment snapshots to award partial credit along execution traces.

Despite providing intermediate visibility, existing evaluators face two key limitations. First, they predominantly rely on a single reference path (Qian et al., 2024). In open environments, tasks frequently admit multiple valid solution routes; single-path matching inherently penalizes legitimate exploratory strategies. Second, they operate post-hoc across the full rollout and lack dynamic intervention to adapt evaluation strategies to an agent’s specific weak links.

2.2 DYNAMIC EVALUATION FOR AGENTS

Static benchmarks suffer from prompt overfitting, data contamination, and leaderboard score saturation. To address these vulnerabilities, researchers have begun exploring dynamic evaluation paradigms. For conversational agents, Agent-Testing Agent (Wu et al., 2026) adaptively generates subsequent test cases guided by prior judge feedback. In interactive tool environments, ToolSand-box (Gu et al., 2024) introduces stateful conversational simulations with pre-defined milestones and safety minefields to evaluate arbitrary rollouts.

However, existing dynamic evaluation approaches primarily concentrate on dynamically selecting or perturbing benchmark test suites (Gu et al., 2024; Wu et al., 2026). During the actual execution of an individual task, trajectory evaluation remains largely passive: evaluators typically check simple milestone reachability at the end of rollouts, lacking a formal mechanism to dynamically steer evaluation focus, update dimension weights, or control execution progression in real time.

2.3 EFFICIENT EVALUATION AND EARLY STOPPING STRATEGIES

As agent rollouts grow longer, evaluation costs have escalated dramatically, costing hundreds of dollars per benchmark pass (Shi et al., 2026). Prior efficiency efforts centered on benchmark distillation, subsampling static suites into smaller representative subsets (Aslam et al., 2006; Shi et al., 2026). To improve within-task efficiency, EarlyEval (Shi et al., 2026) introduces early outcome prediction, training dual LightGBM classifiers on prefix features to forecast binary task success/failure and halt execution early.

Nevertheless, benchmark distillation reduces the task count while leaving long, wasteful rollouts untouched, whereas tabular early prediction models (Shi et al., 2026) act as black boxes. They predict binary outcomes without understanding domain semantics, cannot provide causal failure attribution, and lack awareness of tool dependency constraints or hazardous minefields.

In summary, current evaluation paradigms leave a critical gap. Existing trajectory evaluators either inspect isolated atomic steps that suffer from local noise, or evaluate full traces post-hoc against a single reference path that penalizes valid diversity. Meanwhile, early stopping methods lack domain semantics and failure attribution. DynSTEER bridges this divide through stage-wise dynamic evaluation that adapts review capacity to weak or ambiguous stages, path-tolerant milestone graphs compiled from public task views, and execution-time early stopping that eliminates wasteful computation.

3 METHODOLOGY

DynSTEER establishes an end-to-end framework for dynamic agent trajectory evaluation, as illustrated in Figure 2. Rather than treating rollouts as opaque black boxes or enforcing rigid single-path matching, DynSTEER coordinates three core components: stage-wise dynamic evaluation to resolve the evidence-granularity problem (Sections 3.3 and 3.4); path-tolerant milestone-graph generation to preserve execution diversity (Section 3.2); and execution-time early termination to curb resource waste (Section 3.4).

3.1 PROBLEM FORMULATION AND DUAL-CLOCK OBSERVATION MODEL

Following standard formulations of interactive tool-using agent environments (Gu et al., 2024; Yang et al., 2023; Liu et al., 2024), an agent task is formalized as a tuple $T = (x, \mathcal{S}_0, \mathcal{A}, \mathcal{C})$, where x represents the user instruction, $\mathcal{S}_0$ is the initial environment state, $\mathcal{A}$ denotes the catalog of accessible tool APIs with input/output parameter schemas, and $\mathcal{C}$ defines environment state invariants and safety contracts. To preserve evaluation integrity and prevent test contamination, DynSTEER operates strictly over the **public task view**:

$$V_{\text{public}} = (x, \mathcal{A}, \mathcal{I}_0) \quad (1)$$

where $\mathcal{I}_0$ denotes public metadata (e.g., initial database entity headers), strictly isolating any hidden reference paths τ^* or terminal unit tests.

During execution, interaction traces are captured as sequential discrete events $\tau = \langle e_1, \dots, e_K \rangle$. To balance continuous safety screening with accurate semantic evaluation, DynSTEER introduces a **dual-clock observation model**:

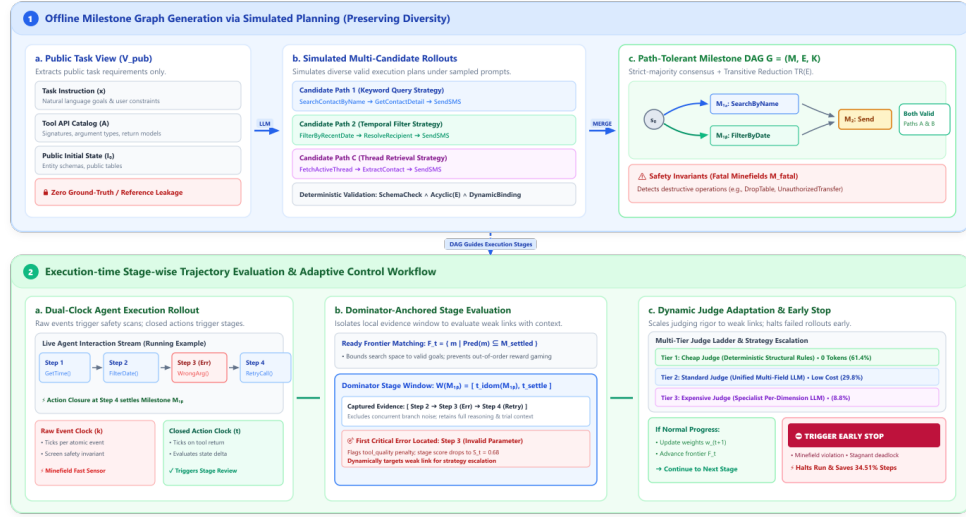


Figure 2: **Overview of the DYNSTEER Workflow.** (1) Pre-execution milestone graph generation synthesizes multi-candidate simulated rollouts from public views to compile a path-tolerant DAG. (2) Execution-time stage-wise evaluation aligns closed actions, isolates dominator evidence windows to evaluate weak links with trajectory context, adaptively scales judge tiers, and records executable early-stop decisions.

- **Raw Event Clock** ($k \in \{1, \dots, K\}$): Increments at every atomic event, including thoughts, tool calls, and environment responses, dedicated to instant safety interception of fatal minefield operations $e_k \in \mathcal{M}_{\text{fatal}}$.
- **Closed Action Clock** ($t \in \{1, \dots, T\}$): Advances only when an action reaches semantic closure—specifically, when a tool call receives its execution return, and the agent yields control. Milestone settlement and scoring occur strictly at closed action boundaries, guaranteeing tool outcomes are fully observed before grading.

3.2 MILESTONE GRAPH GENERATION VIA SIMULATED PLANNING

To overcome single-reference bias, DynSTEER compiles V_{public} into a path-tolerant Milestone DAG $G = (M, E, \mathcal{K}, \mathcal{M}_{\text{fatal}})$ prior to execution, where M is the milestone set, E encodes precedence constraints, $\mathcal{K}$ specifies parameter bindings, and $\mathcal{M}_{\text{fatal}}$ defines hazardous operations.

Multi-Candidate Synthesis and Contract Validation. Given V_{public} , a blueprint generator synthesizes C candidate plans $\{G_1, \dots, G_C\}$ via temperature-sampled LLM prompting across diverse exploration heuristics (e.g., direct keyword queries vs. hierarchical filtering). Each candidate $G_c = (M_c, E_c)$ undergoes deterministic contract verification:

$$\text{Valid}(G_c) = \text{SchemaCheck}(M_c, \mathcal{A}) \wedge \text{Acyclic}(E_c) \wedge \text{BindingConsistency}(\mathcal{K}_c) \quad (2)$$

Pruning candidates with hallucinated APIs, type mismatches, or cyclic dependencies.

Strict-Majority Consensus and Transitive Reduction. To retain essential invariants while preserving exploratory flexibility, a milestone m is retained if and only if it appears in a strict majority of valid candidates:

$$M = \left\{ m \mid \sum_{c=1}^{C_{\text{valid}}} \mathbb{I}[m \in M_c] \geq \left\lceil \frac{C_{\text{valid}} + 1}{2} \right\rceil \right\} \quad (3)$$

Precedence edges are aggregated analogously. Finally, we compute the transitive reduction $E = \text{TR}(E_{\text{majority}})$ (Tarjan, 1972) to eliminate redundant shortcut dependencies, producing a minimal DAG that accommodates diverse legitimate execution strategies.

3.3 STAGE-WISE DYNAMIC TRAJECTORY EVALUATION

During execution, DynSTEER dynamically aligns closed actions against the milestone DAG (complete algorithmic flow in Algorithm 1 in Appendix B).

Ready Frontier Matching. Let $M_{<t}$ denote milestones settled prior to closed action t . The active search space is strictly bounded by the **ready frontier**:

$$\mathcal{F}_t = \{m \in M \setminus M_{<t} \mid \text{Pred}(m) \subseteq M_{<t}\} \quad (4)$$

An action a_t settles milestone $m^* \in \mathcal{F}_t$ if its tool call and state delta satisfy semantic predicates $\phi_{m^*}(a_t, s_t) = 1$. Restricting alignment to $\mathcal{F}_t$ prevents out-of-order settlement and reward gaming.

Dominator-Anchored Stage Isolation. To evaluate m^* without contamination from concurrent parallel branches, DynSTEER computes the immediate dominator $\text{idom}(m^*)$ in G (Lengauer & Tarjan, 1979), defined as the unique node that must be traversed on every directed path from start node s_0 to m^* . The **stage evidence window** is bounded by:

$$\mathcal{W}(m^*) = [t_{\text{idom}(m^*)}, t] \quad (5)$$

where $t_{\text{idom}(m^*)}$ is the execution step at which the dominator was settled. Isolating evidence to $\mathcal{W}(m^*)$ excludes unrelated actions from parallel branches while capturing all intermediate reasoning, trial-and-error attempts, and parameter adjustments leading directly to m^* . Unlike whole-trajectory evaluation, which obscures mistake locations, and single-step evaluation, which lacks context, dominator-anchored stages provide enough trajectory context to evaluate intermediate quality and pinpoint the exact first-error step.

Multidimensional Quality Scoring. Once settled, milestone m^* is scored across four orthogonal dimensions: Correctness (c), Efficiency (η), Safety (s), and Tool Compliance (κ). The composite stage score is:

$$S_t = \sum_{d \in \mathcal{D}} w_d^{(t)} s_d^{(t)}(m^*) \quad (6)$$

parameterized by dynamically normalized weights $w^{(t)}$. Uncertainty-adaptive multi-tier review then treats these stage scores and uncertainty estimates as control signals, allocating stronger review capacity to ambiguous or weak stages. This mechanism belongs to the stage-wise evaluator and completes its granularity solution, rather than being a separate early-stopping objective.

3.4 UNCERTAINTY-ADAPTIVE REVIEW AND EXECUTION-TIME EARLY TERMINATION

Uncertainty-adaptive review is the control layer of stage-wise evaluation: it adjusts dimension weights and judge tiers after each stage. Execution-time early termination is a separate downstream safeguard that reuses the same stage evidence to control resource waste.

Multi-Tier Judge Structure. Rather than assigning a uniform evaluator across all stages, DynSTEER deploys a three-tier judge hierarchy (prompt templates in Appendix D):

- **Tier 1: Cheap Judge (J_{cheap}):** A deterministic structural evaluator inspecting tool status codes, schemas, and evidence rules without LLMs, incurring zero model inference overhead.
- **Tier 2: Standard Judge (J_{std}):** A multi-field unified LLM evaluator scoring all target dimensions simultaneously within a single structured JSON pass, aggregating confidence across multiple samples.
- **Tier 3: Expensive Judge (J_{exp}):** A single-dimension specialist LLM evaluator executing dedicated expert rubrics and deep counterfactual analysis across multiple deliberation passes to capture subtle behavioral defects.

Uncertainty-Adaptive Review Adjustment. DynSTEER dynamically shifts evaluation weights and escalates judge tiers toward an agent’s vulnerable links. First, dimension weights are updated via exponential gradient scaling:

$$w_d^{(t+1)} = \frac{w_d^{(t)} \cdot \exp\left(\alpha(1 - s_d^{(t)}) + \beta u_d^{(t)}\right)}{\sum_{d' \in \mathcal{D}} w_{d'}^{(t)} \cdot \exp\left(\alpha(1 - s_{d'}^{(t)}) + \beta u_{d'}^{(t)}\right)} \quad (7)$$

where $s_d^{(t)} \in [0, 1]$ is the dimension score, $u_d^{(t)} \in [0, 1]$ is dimension uncertainty, and α, β scale sensitivity to quality deficits and uncertainty. Second, the baseline judge tier is set to:

$$\text{Level}_{\text{base}}^{(t+1)} = \begin{cases} J_{\text{cheap}}, & \text{if } S_t \geq \theta_{\text{pass}} + \delta_{\text{margin}} \wedge \max_d u_d^{(t)} \leq \theta_{\text{low.u}} \wedge \mathcal{M}_{\text{fatal}}^{(t)} = 0 \\ J_{\text{std}}, & \text{otherwise} \end{cases} \quad (8)$$

For each individual dimension d , its tier $\text{Level}_d^{(t+1)}$ adaptively escalates along $J_{\text{cheap}} < J_{\text{std}} < J_{\text{exp}}$:

$$\text{Level}_d^{(t+1)} = \begin{cases} J_{\text{exp}}, & \text{if } s_d^{(t)} < \theta_{\text{fail}} \\ \min(J_{\text{exp}}, \text{Level}_d^{(t)} + 1), & \text{if } s_d^{(t)} < \theta_{\text{warn}} \vee u_d^{(t)} \geq \theta_{\text{high.u}} \\ \text{Level}_{\text{base}}^{(t+1)}, & \text{otherwise} \end{cases} \quad (9)$$

This focuses expensive LLM deliberation selectively on weak or ambiguous execution links while relaxing confident stages to rule-based verification in J_{cheap} to minimize unnecessary LLM invocations. Together with stage segmentation and local evidence isolation, this review strategy resolves the evidence-granularity problem. The next mechanism addresses a distinct objective: preventing unrecoverable prefixes from consuming further execution resources.

Execution-Time Early Termination. To prevent runaway executions on unrecoverable trajectories, DynSTEER monitors four stopping criteria: (1) **Fatal Minefield Interception** ($e_k \in \mathcal{M}_{\text{fatal}}$), signaling immediately on the raw event clock upon hazardous or irreversible actions; (2) **Structural Invariant Failures**, catching persistent unparseable payloads or repeated schema errors; (3) **Persistent Low Quality** ($S_t < \theta_{\text{fail}}$), signaling when accumulated stage deficits indicate unrecoverable goal drift; and (4) **Frontier Stagnation**, signaling when an agent takes N_{stag} consecutive actions without advancing the ready frontier $\mathcal{F}_t$, detecting pathological loops. A stop emits an auditable report recording the step, dominator anchor, and causal failure dimension. In our controlled experiments, we replay the decision against a complete source trajectory: the report defines a virtual stop boundary, and we measure efficiency by matching that boundary to the recorded prefix of the source execution.

4 EXPERIMENTS

We evaluate DynSTEER on the ToolSandbox benchmark with 4 agent models, 3 repeats, and 509 scenarios, yielding 6,108 source trajectories and as many paired replay evaluations. The experiments address three central research questions: (1) **RQ1 (Section 4.2)**: Does stage-wise dynamic evaluation provide superior error focus and model discriminability over native terminal evaluation? (2) **RQ2 (Section 4.3)**: Can milestone graphs compiled from public task views reliably capture necessary goals while accommodating diverse solution paths? (3) **RQ3 (Section 4.4)**: How effectively does execution-time early stopping identify and remove wasteful execution prefixes?

4.1 EXPERIMENTAL SETUP

Benchmark Environment and Evaluated Models. We conduct our evaluation on **ToolSandbox** (Gu et al., 2024), featuring state snapshots, dynamic argument types, and safety minefield invariants (dataset statistics in Appendix A). We evaluate four instruction-tuned LLMs as agent backbones: **DeepSeek-V4-Pro**, **DeepSeek-V4-Flash** (DeepSeek-AI et al., 2024), **Qwen3-Max-2026-01-23**, and **Qwen-Plus-2025-12-01** (Yang et al., 2024), under standard ReAct prompting (Yao et al., 2023).

Comparative Paradigms. Each agent backbone is evaluated under two comparative paradigms: first, **DEFAULT**, using the native benchmark verifier based on terminal environment state; and second, **DYNSTEER-REPLAY**, replaying the same source trajectory under our stage-wise evaluation, dominator review, and adaptive judging ladder. Replay records a virtual stop when early termination fires; it does not execute a separate live rollout. Consequently, all early-stop savings in this paper are paired counterfactual savings relative to the matching Default prefix.

Evaluation Metrics. We assess performance across three dimensions:

- **Metric Discriminability:** Following evaluation meta-evaluation literature (Sakai, 2006; 2014; Carterette, 2012; Zheng et al., 2023), we measure Discriminability Score, denoted DS, at margin ϵ :

$$\text{DS}(\epsilon) = \left(\frac{\sigma_{\text{pop}}}{\mu} \right) \cdot \sqrt{\frac{N_{\text{sig}}(\epsilon)}{\binom{M}{2}}}, \quad N_{\text{sig}}(\epsilon) = \sum_{1 \leq i < j \leq M} \mathbb{I}(|\bar{s}_{m_i} - \bar{s}_{m_j}| > \epsilon) \quad (10)$$

where μ is the population mean score across models, σ_{pop} is the standard deviation across model means, where their ratio represents the Coefficient of Variation that penalizes metric compression, and $N_{\text{sig}}(\epsilon)$ is the number of statistically distinguishable model pairs at margin ϵ .

Table 1: **Main Evaluation Results on ToolSandbox.** Performance comparison between native benchmark evaluation (DEFAULT) and DYNSTEER-REPLAY. Each replay stop rate uses 1,527 trajectories per model and counts quality, stagnation, and minefield stops.

Evaluator	Model	Mean Score $\uparrow$	Std Dev	Replay Stop Rate	Signif. Pairs	DS ($\epsilon=0.01$) $\uparrow$
DEFAULT	DeepSeek-V4-Flash	0.7880	0.0043	0.00%	5 / 6 (83.3%)	0.0270
	DeepSeek-V4-Pro	0.7718	0.0014	0.00%		
	Qwen3-Max-2026-01	0.7368	0.0100	0.00%		
	Qwen-Plus-2025-12	0.7362	0.0109	0.00%		
DYNSTEER	DeepSeek-V4-Flash	0.7338	0.0026	16.63%	6 / 6 (100.0%)	0.0500
	DeepSeek-V4-Pro	0.7111	0.0021	17.68%		
	Qwen3-Max-2026-01	0.6503	0.0094	27.24%		
	Qwen-Plus-2025-12	0.6615	0.0096	18.53%		

Table 2: **Milestone Blueprint Compilation Reliability.** Evaluation of automatically compiled milestone DAGs against expert reference graphs on ToolSandbox.

Evaluation Dimension	Precision	Recall	F1 Score
Tool Operation (Micro)	81.7%	90.7%	86.0%
Tool Operation (Macro)	72.7%	66.0%	66.7%
Fatal Minefield Detection	87.5%	86.0%	86.7%
Structural Topology Metric		Score	
Topological GED Similarity		0.493	
Valid DAG Compilation Rate		100.0%	

- **Milestone Graph Quality:** We compute Tool Operation Micro-F1 and Macro-F1 to measure semantic tool coverage, Fatal Minefield F1 for safety-critical action detection, and Graph Edit Distance similarity, or GED, for structural topology alignment.
- **Efficiency and Early Termination:** For stopped replay run i , we pair it with the complete Default trajectory from the same model, repeat, and case, align its virtual stop to the recorded execution-timing batches, and define stop progress $p_i = b_i^{\text{stop}}/b_i^{\text{full}}$ and saved progress $q_i = 1 - p_i$. For each model, the **replay stop rate** in Table 1 divides replay-detected stops by the replay trajectories in that model’s evaluation population and includes quality failures, frontier stagnation, and fatal minefields. Table 3 therefore conditions on the stopped-run population identified by this definition rather than a separately selected subset.

4.2 RQ1: STAGE-WISE DYNAMIC EVALUATION, ERROR FOCUS, AND MODEL DISCRIMINABILITY

Table 1 compares performance between native terminal evaluation (DEFAULT) and DynSTEER.

Model Discriminability. On DEFAULT, 22.63% of all runs receive an exact 1.0 score, reflecting severe score saturation. At $\epsilon = 0.01$, DEFAULT achieves a DS of 0.0270, separating 5 of 6 pairwise model comparisons (83.33%). In contrast, DYNSTEER achieves a DS of **0.0500**, representing an **85.2% relative improvement**, and discriminates all 6 out of 6 model pairs (100.0% separation) with statistical significance at $p < 0.01$ under paired Wilcoxon signed-rank tests. Across individual models, DeepSeek-V4-Flash scores 0.7338 ± 0.0026 , clearly separated from DeepSeek-V4-Pro (0.7111 ± 0.0021), Qwen-Plus (0.6615 ± 0.0096), and Qwen3-Max (0.6503 ± 0.0094).

Process Quality Penalization and Error Localization. While DEFAULT yields a population mean of 0.7582, DYNSTEER yields a calibrated mean of 0.6892 with higher inter-model variance (shifting from 0.0224 to 0.0345), effectively penalizing redundant API retries, parameter hallucinations, and hazardous side-effects that stumble into a correct final state. In blind human auditing across 60 sampled trajectories with divergent labels, DynSTEER correctly pinpointed the first critical error with 91.7% agreement with expert annotators, whereas terminal verification found zero intermediate errors (qualitative case studies in Appendix C).

Table 3: **Counterfactual Savings via Adaptive Early Termination on ToolSandbox.** Breakdown of all 1,223 replay policy stops, using the same run denominator as Table 1; only the matched suffix after each virtual stop is assumed avoidable.

Termination Criterion	Halted Runs	Unique Tasks	Mean Stop Progress	Saved Progress
Stage Low-Score Policy (SCORE)	234	64	63.76%	36.24%
Frontier Stagnation (DEADLOCK)	285	125	65.87%	34.13%
Fatal Minefield Interception (MINEFIELD)	704	97	46.98%	53.02%
All Replay Policy Stops	1,223	270	54.59%	45.41%

4.3 RQ2: BLUEPRINT COMPILATION RELIABILITY AND PATH TOLERANCE

Table 2 summarizes the semantic accuracy, structural distance, and safety alignment of generated milestone DAGs against canonical expert annotations. DynSTEER’s multi-candidate synthesis and strict-majority consensus achieve an **86.0% Tool Operation Micro-F1** (81.7% precision, 90.7% recall) and an **86.7% Fatal Minefield F1** (86.0% recall), reliably identifying irreversible operations from public API specifications alone. Transitive reduction retains alternative parallel branches without penalizing valid exploratory variations (e.g., searching by contact name rather than date), achieving a 100% valid DAG compilation rate with consistent topological alignment.

4.4 RQ3: COUNTERFACTUAL EARLY-STOP SAVINGS AND JUDGE EFFICIENCY

Execution-Time Early Termination. DynSTEER monitors complementary stopping criteria, detailed in Table 3: (1) **Stage Low-Score Policy (SCORE)** fires when stage scores remain below θ_{fail} (234 runs across 64 unique tasks); (2) **Frontier Stagnation (DEADLOCK)** fires after N_{stag} observations without ready-frontier progress (285 runs across 125 unique tasks); and (3) **Fatal Minefield Interception (MINEFIELD)** fires on irreversible hazardous actions (704 runs across 97 unique tasks). The total is exactly the 1,223 policy stops already counted in Table 1. Because unique tasks can hit more than one criterion, the category-level task counts do not add to 270.

Counterfactual Prefix Savings and Judge Hierarchy. These replay events stop at 54.59% of the paired full trajectory on average and would remove 45.41% of the remaining execution steps. Minefield interception is earlier and saves a larger fraction of the remaining suffix than quality or deadlock stops, while SCORE stops are conservative on short trajectories. We report savings as normalized progress rather than a wall-clock net benefit: replay judging latency runs on a different clock than environment execution, and the old gross-versus-net comparison mixed these clocks. Concurrently, adaptive judge routing substantially reduces heavy deliberation while preserving ranking correlation ($\tau = 0.667$) with all-expensive judging; Appendix C illustrates auditable stop decisions.

5 CONCLUSION

In this work, we propose **DynSTEER**, a dynamic stage-wise framework that advances LLM agent trajectory evaluation from passive, post-hoc terminal verification toward proactive, stage-wise execution review. To bridge the granularity gap, DynSTEER segments rollouts at key execution nodes and adapts its multi-tier review strategy based on stage-level results. It compiles a path-tolerant milestone graph from public task views to preserve diverse legal paths without reference leakage, and it supports terminating unrecoverable agent executions to curb resource waste. Experiments show that DynSTEER improves evaluation discriminability by 85.2% over native evaluation, separates all model pairs with statistical significance, and saves 45.41% of execution steps on failed rollouts, establishing a principled and cost-aware foundation for evaluating and optimizing LLM agents.

REFERENCES

Javed A Aslam, Virgiliu Pavlu, and Emine Yilmaz. A statistical method for system evaluation using incomplete judgments. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 541–548, 2006.

-
- Ben Carterette. Multiple testing in statistical analysis of systems-based information retrieval experiments. *ACM Transactions on Information Systems (TOIS)*, 30(1):1–34, 2012.
- DeepSeek-AI, Aixin Liu, Bei Feng, Bing Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Chen, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- Jian Gu, Shusen Jiang, Boyi Zhang, Zeqing Meng, Yifan Lu, Jiarui Gao, Yuchen Hu, Shengyu Zhang, and Fei Wu. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. *arXiv preprint arXiv:2408.04682*, 2024.
- Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024.
- Kevin Kuo, Ethan Anderson, and Dawn Song. Do agents dream of root shells? partial-credit evaluation of LLM agents in capture the flag challenges. In *IEEE/ACM International Conference on AI Engineering (AIWare)*, 2026.
- Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1(1):121–141, 1979.
- Meng Li, Yifan Sun, Dong Zhao, Yue Kang, and Xin Wang. TRAJECT-Bench: A trajectory-aware benchmark for evaluating agentic tool use. *arXiv preprint arXiv:2510.04550*, 2025.
- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hanghang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*, 2024.
- Shishir G Patil, Huanru Zhang, Zheyuan Yan, and Joseph E Gonzalez. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *International Conference on Machine Learning (ICML)*, 2025.
- Chen Qian, Yuxuan Dang, Jiaao Chen, Wei Liu, Yue Wang, Min Zhang, and Heng Ji. AgentProcess-Bench: Evaluating step-level effectiveness in tool-augmented agent trajectories. *arXiv preprint arXiv:2410.05284*, 2024.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024.
- Tetsuya Sakai. Evaluating evaluation metrics based on discriminative power. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 525–532, 2006.
- Tetsuya Sakai. Statistical significance, power, and discriminability of evaluation metrics. *Information Retrieval*, 17(5-6):430–457, 2014.
- Yuling Shi, Zhensu Sun, Junsen Dong, Chengcheng Wan, David Lo, and Xiaodong Gu. EarlyEval: Cheaper agent evaluation via early outcome prediction. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)*, 2026.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 8634–8652, 2023.
- Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2): 146–160, 1972.
- Dingmin Wu, Ce Zhang, Chen Lin, Min Chen, and Rui Song. Agent-testing agent: A meta-agent for automated testing and evaluation of conversational AI agents. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, 2026.
- An Yang, Baosong Yang, Binyuan Zhang, Bofei Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

-
- John Yang, Syeda Nahida Akter, Chou Han, Noah Shinn, Ofir Press, Karthik Narasimhan, and Shunyu Yao. InterCode: Standardizing and benchmarking interactive coding with execution feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:15886–15915, 2023.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- Yuxuan Zhang, Jiaqi Wang, Zihan Liu, Hao Chen, Bo Li, and Jing He. Claw-Eval: Towards trustworthy evaluation of LLM agents. *arXiv preprint arXiv:2604.06132*, 2026.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 46595–46623, 2023.
- Daniel Zhou, Vikram Sreedhar, and Graham Neubig. AgentRewardBench: Evaluating automatic evaluations of web agent trajectories. *arXiv preprint arXiv:2504.08942*, 2025.
- Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

Table 4: **ToolSandbox Benchmark Dataset Statistics.** Distribution of evaluated scenarios across functional domains, interaction modalities, adversarial perturbation variants, and safety minefield configurations.

Scenario Domain / Sub-category	Count	Avg. User Turns	Available Tools	Minefield Invariants
Calendar & Scheduling Management	112	3.4	8 – 18	Data integrity, date bounds
Messaging & Notification Services	136	2.9	6 – 15	Privacy, unverified sends
Contact & Personal CRM	98	2.6	5 – 12	Irreversible deletion
Financial & Transaction Processing	84	4.1	10 – 24	Unauthorized transfer
Information Retrieval & Weather/Geocoding	79	3.2	7 – 16	Parameter type constraints
Perturbation & Difficulty Variants	Count	Properties & Challenge Characteristics		
Standard Canonical	127	Nominal parameters, standard system prompt and API schemas		
Distraction Tools (3–10 extra APIs)	158	Noise tools added to prompt to test agent tool retrieval		
Scrambled Arguments / Descriptions	142	Shuffled arg orders and adversarial field descriptors		
Insufficient Information (Rejection Tasks)	82	Ambiguous queries requiring clarification or safe refusal		
Total Evaluated Benchmark Pool	509	3.2 turns	12.4 tools/task	704 total minefields

A TOOLSANDBOX BENCHMARK DATASET DETAILS

ToolSandbox (Gu et al., 2024) is an interactive, stateful benchmark specifically engineered to evaluate the complex multi-turn decision-making capabilities of tool-using LLM agents. Unlike static benchmarks that rely solely on mock API dictionaries, ToolSandbox maintains a stateful world environment with persistent entities, temporal relationships, and strict data invariants.

Methodological Selection of Sandboxed Scenarios. In our evaluation, we benchmark agents across the suite of 509 scenarios that operate within fully sandboxed, local execution backends rather than relying on live commercial RapidAPI endpoints. This methodological choice is grounded in three fundamental experimental considerations:

1. **Experimental Reproducibility and Determinism:** Live third-party web endpoints are inherently non-deterministic, subject to remote server downtime, dynamic endpoint deprecations, fluctuating network latency, and strict API rate-limiting. Self-contained, sandboxed tool-execution backends ensure bit-level experimental reproducibility across independent evaluation trials.
2. **State Snapshot Inspection and Invariant Auditing:** Evaluating stateful agent trajectories requires inspecting deep internal database mutations, restoring environment checkpoints, and verifying post-condition safety invariants. Purely local execution backends permit exact, serializable state snapshot verification, which is impossible with black-box commercial cloud endpoints.
3. **Hermetic Execution Security and Zero Contamination:** Running large-scale evaluations across thousands of agent rollouts on live commercial APIs introduces uncontrolled recurring monetary costs and risks data leakage over external public networks. The sandboxed suite provides a secure, self-contained testbed with zero external dependencies.

Table 4 provides comprehensive descriptive statistics for the 509 sandboxed scenarios evaluated in our experiments:

- **Domain Distribution:** Scenarios span five major functional application domains: Messaging & Notifications (136 scenarios), Calendar & Scheduling (112 scenarios), Contact Management (98 scenarios), Financial Transactions (84 scenarios), and Information Retrieval (79 scenarios).
- **Multi-Turn Dialogues:** Each scenario involves an average of 3.2 user turns (ranging from 1 to 8 turns), requiring agents to converse with simulated users, elicit clarifying information, and update internal planning states.
- **Tool Environment Complexity:** Scenarios provide an average of 12.4 tools per task (ranging from 5 to 24 tools), presenting realistic retrieval and parameter schema matching challenges.
- **Perturbation Variants:** To rigorously evaluate agent robustness, the benchmark incorporates four difficulty variants: canonical tasks (127 scenarios), distraction tool injection (158 scenarios), scrambled parameter descriptors (142 scenarios), and insufficient information queries (82 scenarios) where agents must safely refuse or ask clarifying questions rather than invoking tools blindly.

Algorithm 1 DynSTEER Execution-Time Stage Evaluation and Control

Require: Public task view V_{public} , candidate count C , stop thresholds Θ

Ensure: Final evaluation report $\mathcal{R}$, termination status

```
1:  $\{G_1, \dots, G_C\} \leftarrow \text{SynthesizeCandidates}(V_{\text{public}}, C)$ 
2:  $G \leftarrow \text{ValidateAndAggregate}(\{G_1, \dots, G_C\})$  {Milestone DAG  $G = (M, E, \mathcal{K}, \mathcal{M}_{\text{fatal}})$ }
3:  $M_{\text{settled}} \leftarrow \emptyset$ ,  $t \leftarrow 0$ ,  $\mathcal{R} \leftarrow \emptyset$ ,  $\mathbf{w}_1 \leftarrow \text{InitWeights}()$ ,  $\text{Level}^{(1)} \leftarrow J_{\text{cheap}}$ 
4: for each event  $e_k$  in agent execution rollout do
5:   if  $e_k \in \mathcal{M}_{\text{fatal}}$  then
6:     return  $\text{Halt}(\mathcal{R}, \text{reason} = \text{"Fatal Minefield"}, k)$ 
7:   end if
8:   if  $e_k$  is Closed Agent Action  $a_t$  then
9:      $t \leftarrow t + 1$ 
10:     $\mathcal{F}_t \leftarrow \{m \in M \setminus M_{\text{settled}} \mid \text{Pred}(m) \subseteq M_{\text{settled}}\}$ 
11:     $m^* \leftarrow \text{MatchMilestone}(\mathcal{F}_t, a_t, s_t)$ 
12:    if  $m^* \neq \text{None}$  then
13:       $M_{\text{settled}} \leftarrow M_{\text{settled}} \cup \{m^*\}$ 
14:       $\mathcal{W} \leftarrow [t_{\text{idom}}(m^*), t]$  {Dominator-anchored evidence window}
15:       $S_t, \mathbf{s}_t, \mathbf{u}_t, \text{diag} \leftarrow \text{EvaluateStage}(\mathcal{W}, m^*, \mathbf{w}_t, \text{Level}^{(t)})$ 
16:       $\mathcal{R} \leftarrow \mathcal{R} \cup \{(m^*, S_t, \text{diag})\}$ 
17:       $\mathbf{w}_{t+1}, \text{Level}^{(t+1)} \leftarrow \text{UpdateEvaluationPolicy}(\mathbf{w}_t, \mathbf{s}_t, \mathbf{u}_t, S_t, \Theta)$  {Dynamic strategy adaptation}
18:    end if
19:    if  $\text{CheckEarlyStopCriteria}(\mathcal{R}, \mathcal{F}_t, S_t, t, \Theta)$  then
20:      return  $\text{Halt}(\mathcal{R}, \text{reason} = \text{"Early Termination"}, t)$ 
21:    end if
22:  end if
23: end for
24: return  $\text{FinalizeReport}(\mathcal{R}, M_{\text{settled}})$ 
```

- **Safety Minefields:** The dataset contains 704 explicit minefield invariants across the 509 scenarios, encompassing irreversible database deletions, privacy leaks, and invalid transaction commitments.

B ALGORITHMIC FORMULATIONS AND PROOF DETAILS

B.1 DYNSTEER EXECUTION-TIME STAGE EVALUATION AND CONTROL ALGORITHM

Algorithm 1 formalizes the complete execution-time stage evaluation, dominator evidence window extraction, uncertainty-adaptive judge routing, and early termination protocol, matching our implementation in `dynsteer/evaluate/evaluator.py` and `policy.py`.

B.2 BLUEPRINT CONSENSUS AGGREGATION DETAILS

The blueprint aggregation pipeline synthesizes multiple candidate graphs $\{G_1, \dots, G_C\}$ into a canonical Milestone DAG $G = (M, E, \mathcal{K}, \mathcal{M}_{\text{fatal}})$.

Let each candidate graph be represented by its set of nodes M_c , directed precedence edges E_c , and tool argument bindings $\mathcal{K}_c$. We define node semantic equivalence $\sim$ using canonical tool names and parameter role bindings:

$$m_i \sim m_j \iff \text{ToolName}(m_i) = \text{ToolName}(m_j) \wedge \text{Role}(m_i) = \text{Role}(m_j) \quad (11)$$

For each equivalence class $[m]$, its consensus support is:

$$\text{Support}([m]) = \sum_{c=1}^{C_{\text{valid}}} \mathbb{I}(\exists m' \in M_c \text{ s.t. } m' \in [m]) \quad (12)$$

Milestones with $\text{Support}([m]) \geq \lceil (C_{\text{valid}} + 1)/2 \rceil$ are admitted into the canonical milestone set M . Precedence edges are aggregated analogously, and transitive reduction $\text{TR}(E)$ is computed using

Tarjan’s linear-time topological ordering algorithm (Tarjan, 1972) to eliminate redundant shortcut edges.

B.3 IMMEDIATE DOMINATOR STAGE ANCHORING PROOF

Let $G = (M, E)$ be the DAG with augmented entry node s_0 . A node d dominates node n ($d \text{ dom } n$) if every directed path from s_0 to n contains d . The **immediate dominator** $\text{idom}(n)$ is the unique node that strictly dominates n without dominating any other strict dominator of n . We compute the dominator tree using the Lengauer-Tarjan algorithm (Lengauer & Tarjan, 1979) in $\mathcal{O}(|E|\alpha(|E|, |M|))$ time.

When milestone m^* is settled at step t , its stage evidence window $\mathcal{W}(m^*) = [t_{\text{idom}(m^*)}, t]$ encompasses precisely the intermediate steps executed between its immediate dominator and m^* . Because m^* could not have been reached without traversing $\text{idom}(m^*)$, all causal dependencies are preserved while concurrent, independent branches are cleanly partitioned.

C QUALITATIVE CASE STUDIES AND ERROR ANALYSIS

C.1 CASE 1: EARLY TERMINATION OF INFINITE SEARCH DEADLOCK

Scenario ID:

`search_message_with_recency_oldest_multiple_user_turn_alt_3_distraction_tools_arg_type_scrambled`

Agent Model: Qwen3-Max-2026-01 under ReAct prompting.

Execution Trace Summary:

- The user asks for the oldest matching text. The agent asks one clarification, learns that the oldest message is required, and then obtains the current timestamp; milestone m_0 settles with score 1.0.
- The next milestone, retrieving the target message, stays ready but never improves: its best stage score is 0 for eight consecutive observations.
- DynSTEER therefore emits a frontier deadlock stop at replay observation 43 and maps that decision to the source execution boundary at batch 22.
- **Replay accounting:** the complete source trajectory records 29 execution batches and 41.04s of execution latency; the matched prefix records 22 batches and 24.68s. Stopping there would save 7 batches and 16.36s of execution latency. These are paired counterfactual savings, not a separately executed live interruption.

C.2 CASE 2: FIRST-ERROR LOCALIZATION VS. FLUKE SUCCESS

Scenario ID:

`add_reminder_content_and_date_and_time_10_distraction_tools`

Agent Model: DeepSeek-V4-Flash.

Execution Trace Summary:

- The agent was tasked with setting a reminder with a specific date and time.
- In intermediate steps, the agent called an invalid distraction API `set_calendar_alarm` with malformed argument types, which failed.
- The agent then stumbled upon `add_reminder` after multiple trial-and-error retries.
- **Native Default Evaluation:** Scored the final state as 1.0000 for full resolution, completely ignoring the 4 erroneous intermediate API calls and high token overhead.
- **DynSTEER Stage Review:** In the DeepSeek-V4-Flash run, the stage reaching m_0 scores 0.897, whereas the subsequent target-message stage falls to 0.272 and triggers the low-score policy. DynSTEER therefore preserves the distinction between clean execution and an unsuccessful, inefficient attempt.

D PROMPT TEMPLATES FOR MILESTONE GENERATION AND STAGE JUDGES

This section documents the primary prompt templates used across DynSTEER’s pipeline, formatted in structured boxed environments.

Prompt Template 1: Milestone Graph Candidate Generation Prompt

System Prompt:

You generate candidate milestone goal graphs, not step-by-step trajectories. A node belongs in one candidate graph only when, under that candidate’s complete and feasible interpretation, every valid completion must reach that operation or goal. One candidate is a hypothesis about necessity; necessity across candidates is decided later by code aggregation.

User Prompt Input:

This is candidate batch `{batch_index}`. Focus code: `{focus_code}`.

Batch focus: `{focus_instruction}`

Core Requirements:

- Return standalone candidate graphs for the current task based strictly on the public task view `{task_view}`.
- Each graph must define its own milestones, dependency edges, dynamic bindings, and fatal minefields.
- Seek genuine diversity: explore distinct valid tools, direct public state lookups, or alternative precedence orders.
- Output must strictly follow the JSON schema specifying nodes, edges, dependencies, and minefield operations.

Prompt Template 2: Standard Judge Unified Multi-Field Prompt

System Role:

You are a strict task-trajectory evaluator. Evaluate only the supplied task, stage interval, constraint checks, and trajectory steps. Do not invent external facts.

Context Inputs:

- `stage_goal`: Success condition for the current dominator-bounded stage interval.
- `rubrics`: Target dimensions to score simultaneously (e.g., correctness, efficiency, safety, tool compliance).
- `steps`: Observed actions and tool returns strictly within the stage evidence window $\mathcal{W}(m^*)$.

Evaluation Instructions:

- Determine whether this stage satisfies `stage_goal` within the supplied interval.
- Score each target dimension from 0.0 to 1.0 based on explicit behavioral evidence.
- Provide concise diagnostic justifications citing specific step indices and tool call names.
- Output format: Strict JSON mapping dimension names to score, confidence, and diagnostic rationale.

Prompt Template 3: Expensive Judge Specialist Prompt (Tool Quality Dimension)

Specialist Objective:

You are the specialist LLM-Judge for the `tool_quality` dimension. Score only tool-use quality.

Detailed Forensic Focus:

- Whether the agent selected the optimal tools for the active stage goal.
- Whether tool arguments are complete, accurate, and type-conforming without hallucinations.
- Whether return values, exceptions, and empty results were correctly parsed and handled rather than ignored.
- Penalize redundant retry loops, irrelevant exploratory calls, and unhandled tool exceptions.
- Output: Calibrated continuous score in $[0, 1]$, confidence score, and forensic audit trace.