ADAPTIVE-GROUP PROCESS REWARDS FOR GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training graphical user interface (GUI) agents with reinforcement learning suffers from sparse outcome rewards, leading to inefficient learning. Process rewards offer a promising direction for credit assignment to intermediate actions; however, accurately evaluating an action requires identifying evidence across long-horizon GUI trajectories, as its value may depend on information beyond its immediate state and observation. In this paper, we propose an adaptive-group process reward framework that dynamically partitions a GUI trajectory into semantically coherent subgoal groups, enabling process reward estimation at both the action and group levels. At the action level, a critic evaluates each action based on the visual changes and subsequent actions within the group for action correctness and contribution to the subgoal. At the group level, the framework aggregates action-level evaluations to assess subgoal completion and contribution to the overall task, and further refines these assessments by leveraging the context of all groups in the trajectory. The process reward of each action is obtained by combining its action-level evaluation with the refined group-level evaluation. We further augment GRPO's outcome advantage with the proposed process rewards to provide finer-grained guidance for GUI agent training. Experiments on process reward benchmarks and downstream RL training demonstrate the effectiveness of the proposed framework.

1 INTRODUCTION

Recent advances in large language models and vision-language models have enabled graphical user interface (GUI) agents Cheng et al. (2024); Hong et al. (2024); Qin et al. (2025); Lin et al. (2024); Xu et al. (2025); Gao et al. (2024); Zhang et al. (2025). These agents translate user instructions and visual observations into executable interactions across digital devices, with applications such as software testing, office assistance, and industrial automation. GUI tasks typically involve long-horizon, multi-turn interactions Zhou et al. (2023); Rawles et al. (2024); Xie et al. (2025), requiring agents to maintain multimodal interaction context across changing visual states and coordinate sequences of actions. Reinforcement learning (RL) Li et al. (2025); Xiao et al. (2025); Lu et al. (2025); Tang et al. (2026) is well suited to such interactive tasks, allowing agents to improve their policies based on feedback from the GUI environments.

Most GUI environments provide reward signals primarily in the form of sparse outcome rewards Xie et al. (2025); Li et al. (2025): a trajectory is judged successful if the final application state satisfies the instruction requirement, and unsuccessful otherwise. Consequently, the outcome-level signal is assigned to all intermediate actions in a trajectory, failing to distinguish useful actions from redundant or incorrect ones and making credit assignment unreliable. Process rewards Lightman et al. (2023); Wang et al. (2023); Chen et al. (2025) are therefore needed to evaluate intermediate actions. However, constructing reliable process rewards for GUI trajectories is challenging because GUI environments rarely provide direct supervision for intermediate actions. Whether an action is correctly executed can often be verified from the current and subsequent observations, whereas whether its semantic purpose is correct may only become clear after subsequent actions. Judging the action requires evidence from a broader trajectory context than the immediate state change after that action alone.

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

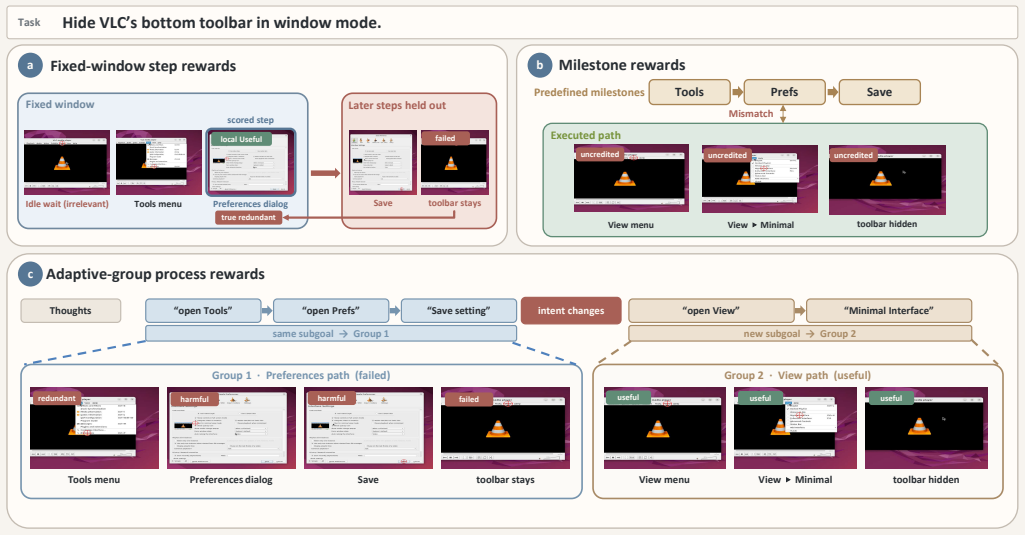

Figure 1: Fixed-window step rewards, milestone rewards, and adaptive-group process rewards on an executed GUI trajectory. (a) A fixed local window can include irrelevant context and miss later evidence. (b) Predefined milestones can leave a valid executed path uncredited. (c) Adaptive groups follow the executed subgoals and score actions within each group.

Existing GUI process reward methods typically organize trajectory evidence and assign credit through two types of preset context scopes: fixed-window step rewards Chen et al. (2025); Xiong et al. (2025); Xiao et al. (2025); Lu et al. (2025); Wan et al. (2026) and milestone rewards Zheng et al. (2026); Hao et al. (2026); Li et al. (2026a); Dai et al. (2026a); Zhai et al. (2026). Fixed-window step rewards evaluate an action within a fixed local context (figure 1(a)). Short windows may miss long-range evidence needed to assess the action's effect, whereas longer windows may introduce irrelevant context. Milestone rewards instead predefine task progress around fixed checkpoints or subgoals (figure 1(b)). Because GUI tasks may admit multiple valid execution paths, such milestones can under-credit progress that does not follow the predefined path and overlook the intermediate actions connecting checkpoints. Thus, an effective process reward should adapt its evidence scope and credit assignment to the executed trajectory, rather than relying on fixed local scopes or predefined checkpoints.

In this paper, we introduce an adaptive-group process reward framework for GUI trajectories (figure 1(c)). Using the task instruction and the agent's responses, the executed trajectory is dynamically partitioned into consecutive groups of actions that share the same subgoal or semantic operation, rather than by a fixed partition rule. This grouping defines the context for judging the actions in each group and supports two complementary levels of evaluation. At the action level, the critic assesses whether each action is correctly executed based on the local visual change and whether it advances the group's subgoal based on later actions in the group. At the group level, the critic aggregates the action-level assessments within each group to assess its subgoal completion and contribution to the whole task. Then, the produced group-level assessments are revisited in the context of all groups, allowing the critic to judge each group's role beyond its local evidence. The process reward of each action is obtained by combining its action-level assessment with the refined group-level assessment of its group. We further use the generated process rewards to shape the GRPO Shao et al. (2024) advantage at each step, providing finer-grained credit assignment during policy optimization. We evaluate the framework on process-reward benchmarks and in downstream RL on OSWorld. The results show consistent improvements in process-reward accuracy and task success rate, demonstrating the effectiveness of adaptive-group process rewards.

Our contributions are as follows:

- We introduce an adaptive-group process reward framework for GUI agents, which dynamically partitions trajectories into semantically coherent subgoal groups to organize the evidence needed for process reward evaluation.

- We design a two-level evaluation that assesses execution and contribution at both the action level and the group level, and obtain the process reward of each action by combining these assessments, making correct execution and group-level contribution jointly necessary.

- Experiments on process reward benchmarks and downstream RL training demonstrate the effectiveness of the proposed framework.

2 Related Work

2.1 GUI Agents

Recent progress in large language models and vision-language models has enabled GUI agents that interact with applications through visual observations and executable actions. Early systems mainly relied on prompting or supervised fine-tuning, while recent open-source models and frameworks, such as ShowUI Lin et al. (2024), CogAgent Hong et al. (2024), AGUVIS Xu et al. (2025), UI-TARS Qin et al. (2025), OpenCUA Wang et al. (2025), Agent S Agashe et al. (2024), and TongUI Zhang et al. (2025), improve grounding, planning, and cross-platform interaction through larger GUI data and stronger vision-language-action modeling. More recently, reinforcement learning has become a central direction for GUI agents because it can optimize long-horizon interaction from environment feedback rather than only imitating fixed demonstrations. DART Li et al. (2025) decouples multi-turn rollout from training for GUI RL, UI-Genie Xiao et al. (2025) builds a reward-guided self-improvement loop for mobile GUI agents, Orcust Lu et al. (2025) combines principle-constrained reward modeling with online VM-grounded trajectory construction, and ClawGUI Tang et al. (2026) provides a full-stack infrastructure for GUI agent RL and deployment. However, most RL pipelines still rely heavily on sparse outcome rewards, which makes it difficult to assign credit to intermediate actions. Our work addresses this limitation by constructing dense process rewards over adaptive groups and the actions in those groups.

2.2 Process Reward Modeling

Process reward modeling supplements sparse outcome rewards with feedback on intermediate steps, enabling more fine-grained credit assignment during training. In reasoning Lightman et al. (2023); Wang et al. (2023) and coding Le et al. (2022); Liu et al. (2023); Dai et al. (2026b) tasks, intermediate feedback can often be verified from execution results, compiler errors, unit tests, or other deterministic checks. GUI tasks provide no comparable automatic signal for most intermediate operations: the value of a click, text entry, menu selection, or file operation depends on visual state changes, task requirements, and subsequent operations. GUI process reward modeling must therefore infer both whether an action is correctly executed and its semantic role from trajectory context. Existing GUI process reward methods can be broadly organized into fixed-window step rewards Chen et al. (2025); Xiong et al. (2025); Xiao et al. (2025); Lu et al. (2025); Wan et al. (2026) and milestone rewards Zheng et al. (2026); Hao et al. (2026); Li et al. (2026a); Dai et al. (2026a); Zhai et al. (2026). Fixed-window step rewards judge actions within a fixed local context. GUI-Shepherd Chen et al. (2025) learns step-level scores and rationales from human annotations, while GUI-PRA Xiong et al. (2025) selects historical context and actively perceives UI state changes to improve step-level judgment. SeekJudge Wan et al. (2026) further targets fine-grained step-level judgments for use within reinforcement learning. Milestone rewards organize feedback around predefined subgoals, checkpoints, or broader task-level structures. ADMIRE Zheng et al. (2026) dynamically distills verifiable milestones from successful explorations and applies asymmetric credit assignment to successful and failed trajectories. StainFlow Hao et al. (2026) tracks entity evidence to derive task stages and links relevant steps for key-node verification, while GUIDE Zhai et al. (2026) decomposes completed trajectories into subtask units for structured diagnosis. These approaches organize action judgment within preset local contexts or task-level structures, whereas our adaptive-group framework dynamically partitions the executed trajectory into semantically coherent subgoal groups and combines execution and contribution at both the action level and the group level to assign each action a process reward.

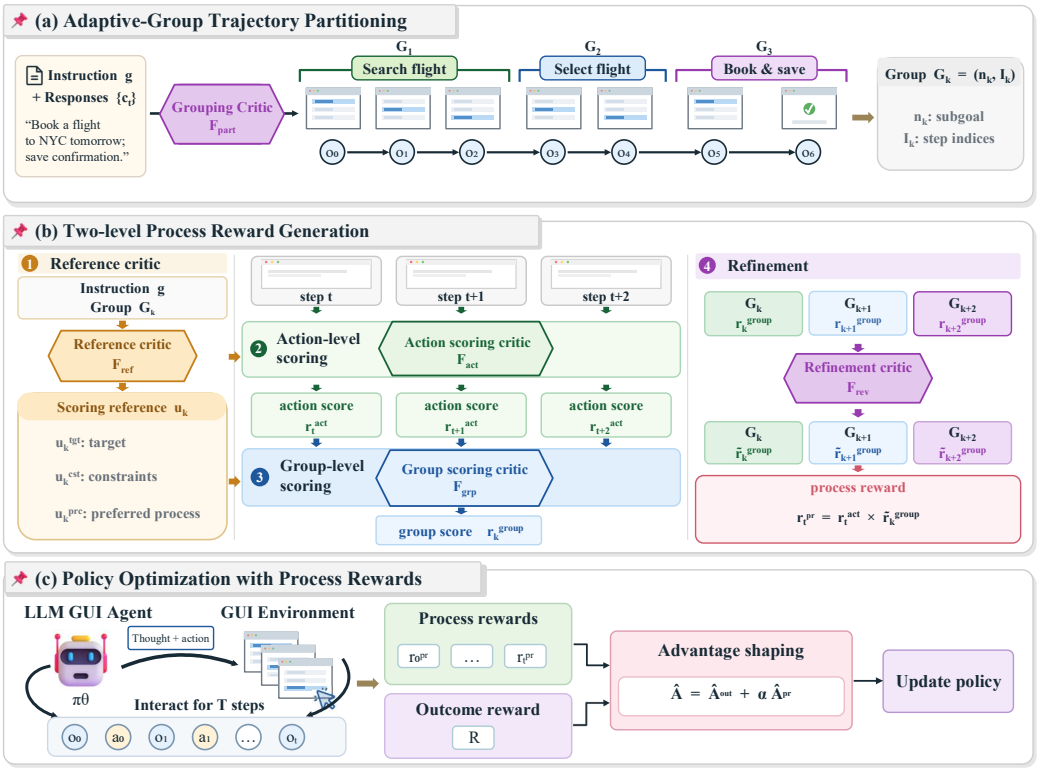

Figure 2: Overview of the proposed framework. (a) A grouping critic partitions the trajectory from the instruction and responses into subgoal groups. (b) Two-level scoring produces an action-level score and a refined group-level score, which are used to generate the process rewards. (c) The process rewards shape the GRPO advantage.

3 METHOD

3.1 PROBLEM FORMULATION

We formulate solving a GUI task as a sequential interaction between an agent and an environment. Given a task instruction g, the agent observes the GUI state o_t at each step. Conditioned on this observation and the interaction history, it generates a response c_t with an executable action a_t. Executing a_t changes the environment and produces the next observation o_{t+1}. An executed trajectory is therefore represented as

$$\tau = (o_0, c_0, a_0, o_1, c_1, a_1, \ldots, o_{T-1}, c_{T-1}, a_{T-1}, o_T), \tag{1}$$

where T is the number of executed actions. The response c_t provides the agent's textual intent, while the observations and executed actions record the interaction with GUI.

After the trajectory is completed, the environment or a task verifier V produces an outcome reward $R = V(g, \tau)$. This trajectory-level signal does not directly indicate the value of individual actions. We therefore formulate GUI process reward generation as producing a process reward for every executed action:

$$(r_0^{\mathrm{pr}}, r_1^{\mathrm{pr}}, \ldots, r_{T-1}^{\mathrm{pr}}) = \phi(g, \tau), \tag{2}$$

where ϕ is a process reward model and r_t^{pr} is the process reward of action a_t at step t. As shown in figure 2, we instantiate ϕ by first partitioning τ into adaptive groups that define the context for judging each action (section 3.2), then generating action-level and group-level assessments (section 3.3). The resulting process rewards are used to shape GRPO's step-level advantage during policy optimization (section 3.4).

3.2 ADAPTIVE-GROUP TRAJECTORY PARTITIONING

Fixed-window step rewards evaluate an action within a preset local context, while milestone rewards organize credit around predefined checkpoints. Neither evidence scope is determined by the trajectory that was actually executed. We recover subgoals from the executed trajectory instead. Grouping should follow executed intents, which are already stated in each c_t, so we use a grouping critic F_{part} to partition the trajectory from the instruction g and the responses $\{c_t\}_{t=0}^{T-1}$:

$$\{G_k\}_{k=1}^{K} = F_{\text{part}}(g, \{c_t\}_{t=0}^{T-1}), \qquad G_k = (n_k, \mathcal{I}_k). \tag{3}$$

The result is a contiguous, non-overlapping partition of the trajectory: each G_k consists of a hypothesized subgoal n_k and the step indices $\mathcal{I}_k$ of the group, and $k(t)$ denotes the unique group of step t. The subgoal n_k is not a required milestone of g, so valid executions can induce different groups. These groups are the context for the judgments below.

3.3 ADAPTIVE-GROUP PROCESS REWARD GENERATION

The adaptive group defines the context for judging its actions and supports two complementary levels of evaluation: how an action contributes to the group's subgoal, and how the subgoal contributes to the overall task. At the action level, the critic assesses whether each action is correctly executed and whether it advances the group's subgoal. At the group level, it aggregates these assessments to judge the group's subgoal completion and contribution to the global task, then revisits the group-level assessments using all groups as context. Verifying execution can be done from the current and next observations, whereas judging semantic purpose depends on what follows: later actions in the group for an action, and other groups for a subgoal.

To score an action or a group, the executed actions have to be interpreted against the instruction. We therefore first use a reference critic F_{ref} to build, for each G_k, an instruction-conditioned reference u_k from g, G_k, and the group's responses $\{c_t\}_{t\in\mathcal{I}_k}$, to inform the two levels of scoring that follow:

$$u_k = (u_k^{\text{tgt}}, u_k^{\text{cst}}, u_k^{\text{prc}}) = F_{\text{ref}}(g, G_k, \{c_t\}_{t\in\mathcal{I}_k}). \tag{4}$$

u_k^{tgt} is the target state implied by g for this group, u_k^{cst} the checkable constraints, and u_k^{prc} the preferred process. Action-level scoring consults u_k^{cst} when checking whether the change from o_t to o_{t+1} is consistent with the instruction; group-level scoring consults u_k^{tgt} and u_k^{prc} when judging whether the group advances the task and whether the attempted route is reasonable. We treat u_k as a scoring reference rather than as a strict prior: the trajectory is not required to follow u_k^{prc} or to attain u_k^{tgt}, and an alternative path that still satisfies g remains valid.

Action-level scoring. The actions of G_k are scored together, not one call per step. An action-level scoring critic F_{act} takes g, u_k, G_k, the group's responses $\{c_t\}_{t\in\mathcal{I}_k}$, and the group's screenshots, and returns a judgment for every action in G_k:

$$J_k = \{(e_t, p_t, h_t)\}_{t\in\mathcal{I}_k} = F_{\text{act}}\big(g, u_k, G_k, \{c_t\}_{t\in\mathcal{I}_k}, \{o_t\}_{t\in\mathcal{I}_k\cup\{t_k^+\}}\big), \tag{5}$$

where $t_k^+ = 1 + \max\mathcal{I}_k$ is the observation after the last action of the group, $e_t \in \{\text{Completed}, \text{Partial}, \text{Failed}, \text{Uncertain}\}$ is the execution result, $p_t \in \{\text{Milestone}, \text{Useful-efficient}, \text{Useful-inefficient}, \text{Exploratory}, \text{Redundant}, \text{Harmful}\}$ is the contribution to the group's subgoal, and h_t is a concise evidence summary of the action. Later actions in G_k remain available as post-action context, so p_t can be judged from the subsequent semantic effect rather than from the local state change alone. When the group is long and image-heavy, attaching every screenshot can exceed the context budget. We then split $\mathcal{I}_k$ into successive windows $\mathcal{W} \subseteq \mathcal{I}_k$ and apply the same critic to each window: the call still receives the whole group's responses as text, but screenshots only for $\{o_t\}_{t\in\mathcal{W}\cup\{t_{\mathcal{W}}^+\}}$, where $t_{\mathcal{W}}^+ = 1 + \max\mathcal{W}$. Concatenating the window outputs yields J_k. The two labels of each action are then mapped by a fixed table M to a scalar action-level score $r_t^{\text{act}} = M(e_t, p_t)$ in $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ (Appendix A.1). The structured judgments J_k are passed to group-level scoring.

Group-level scoring. Whether G_k completes its subgoal, and whether the subgoal helps g, cannot be read from any single action. A group-level scoring critic F_{grp} therefore scores the group from the

structured action-level judgments J_k—not the scalars r_t^{act} alone—together with u_k and summaries $H_{<k}$ of preceding groups, which situate G_k in the progress already made. It judges the group by e_k for subgoal completion and p_k for contribution to the task, then maps these labels with the same table M to a group-level score r_k^{group}:

$$(e_k, p_k, h_k) = F_{\text{grp}}(g, u_k, G_k, J_k, H_{<k}), \qquad r_k^{\text{group}} = M(e_k, p_k). \tag{6}$$

h_k is a concise summary of G_k and is appended to $H_{<k}$ for later groups. Groups are scored in execution order, once per G_k.

Group-level refinement. After the initial pass scores groups in execution order, we use a refinement critic F_{rev} to revisit (e_k, p_k) with g and the full sequence of group conclusions, using this more complete context to reassess each group's role and obtain a refined group-level score $\tilde{r}_k^{\text{group}}$:

$$\{(\tilde{e}_k, \tilde{p}_k)\}_{k=1}^{K} = F_{\text{rev}}\big(g, \{G_k, J_k, e_k, p_k\}_{k=1}^{K}\big), \qquad \tilde{r}_k^{\text{group}} = M(\tilde{e}_k, \tilde{p}_k). \tag{7}$$

Process reward aggregation. The process reward of action a_t combines the two levels by multiplying its action-level score r_t^{act} with the refined group-level score of $G_{k(t)}$:

$$r_t^{\text{pr}} = r_t^{\text{act}}\, \tilde{r}_{k(t)}^{\text{group}}. \tag{8}$$

The product makes both assessments jointly necessary: r_t^{pr} is high only if the action is correctly executed and its group contributes to the task. The score mapping M and the critic prompts are given in Appendix A.

3.4 Policy Optimization with Process Rewards

The process rewards r_t^{pr} are used to shape GRPO's advantage at each step. For an instruction g, GRPO Shao et al. (2024) samples N trajectories $\mathcal{B}_g = \{\tau_i\}_{i=1}^{N}$ and standardizes their outcome rewards $\{R_i\}$ within $\mathcal{B}_g$ by the mean μ^{out} and standard deviation σ^{out}:

$$\hat{A}_i^{\text{out}} = \frac{R_i - \mu^{\text{out}}}{\sigma^{\text{out}} + \epsilon}. \tag{9}$$

This term is constant across the steps of τ_i, so useful and redundant actions at those steps receive the same learning signal.

To distinguish these actions, we keep this outcome term and add a process advantage that is standardized over all steps in $\mathcal{B}_g$, following StainFlow Hao et al. (2026). Let μ^{pr} and σ^{pr} be the mean and standard deviation of $\{r_{i,t}^{\text{pr}}\}$ across every step of every trajectory in the sampled group, and let $\alpha > 0$ be a coefficient:

$$\hat{A}_{i,t}^{\text{pr}} = \frac{r_{i,t}^{\text{pr}} - \mu^{\text{pr}}}{\sigma^{\text{pr}} + \epsilon}, \qquad A_{i,t} = \hat{A}_i^{\text{out}} + \alpha \hat{A}_{i,t}^{\text{pr}}. \tag{10}$$

Policy optimization uses $A_{i,t}$ in place of the uniform outcome advantage.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate the proposed process rewards on two process-reward benchmarks and for reinforcement learning on OSWorld applications. CUARewardBench Lin et al. (2026) is built on OSWorld trajectories from 10 desktop software categories and 7 agents; it annotates key good and bad actions, and we report process-reward results compared to the benchmark's OpenCUA-Reflector Wang et al. (2025) and SE-WSM Sun et al. (2025) prompt templates. CUAStepBench Wan et al. (2026) densely annotates every step on executed computer-use trajectories and pairs these labels with human trajectory-level verdicts, covering 177 applications across platforms. OSWorld Xie et al. (2025) provides open-ended GUI tasks in a real desktop environment, with success judged by execution-based scripts on the final state; we train and test on its Chrome, LibreOffice Impress, and LibreOffice Calc applications.

Table 1: Process reward evaluation on CUARewardBench, compared with models under the OpenCUA-Reflector template. P, NPV, R, S, and OA denote precision, negative predictive value, recall, specificity, and overall accuracy. Best results are in bold; second-best are underlined.

Reward model	P	NPV	R	S	OA
Qwen2.5-VL-7B	54.4	49.4	53.8	50.0	52.0
Qwen2.5-VL-32B	60.3	64.8	79.8	41.5	61.7
Qwen2.5-VL-72B	58.5	64.8	83.0	34.8	60.1
GLM-4.5V-106B	64.0	78.5	89.0	44.5	67.9
Qwen3-VL-32B-T	68.3	77.3	85.2	56.1	71.4
Qwen3-VL-235B-T	66.5	75.0	84.1	53.0	69.4
GPT-5	<u>72.7</u>	**87.8**	<u>92.3</u>	<u>61.6</u>	<u>77.7</u>
GUI-Owl-7B	64.6	61.8	67.0	59.1	63.3
GUI-Owl-32B	61.6	61.5	71.4	50.6	61.6
Qwen3.6-27B	62.5	<u>87.0</u>	**95.1**	36.6	67.3
Ours	**79.4**	84.1	87.8	**73.9**	**81.3**

Table 2: Process reward evaluation on CUARewardBench, compared with models under the SE-WSM template. Best results are in bold; second-best are underlined.

Reward model	P	NPV	R	S	OA
Qwen2.5-VL-7B	56.7	54.2	67.0	43.3	55.8
Qwen2.5-VL-32B	65.3	62.4	67.8	59.8	64.0
Qwen2.5-VL-72B	58.7	67.1	85.2	33.5	60.7
GLM-4.5V-106B	69.5	64.2	66.1	67.7	66.9
Qwen3-VL-32B-T	78.2	66.3	63.2	<u>80.5</u>	71.4
Qwen3-VL-235B-T	74.5	63.2	59.3	77.4	67.9
GPT-5	**83.7**	<u>68.8</u>	64.8	**86.0**	<u>74.9</u>
GUI-Owl-7B	56.6	64.2	86.8	26.2	58.1
GUI-Owl-32B	57.4	<u>68.8</u>	**89.0**	26.8	59.5
SE-WSM-7B	58.7	52.0	48.4	62.2	54.9
Qwen3.6-27B	73.4	68.2	69.8	72.0	70.8
Ours	<u>79.4</u>	**84.1**	<u>87.8</u>	73.9	**81.3**

Metrics. CUARewardBench treats each step judgment as a binary decision against a human label. We use a threshold of 0.4 to separate positive and negative predictions. We follow the benchmark protocol: precision (P) and negative predictive value (NPV) measure how reliable a positive or negative verdict is, while recall (R), specificity (S), and overall accuracy (OA) summarize coverage and overall correctness. On CUAStepBench, every step carries a binary human label, and we score the predicted step judgments by precision (P), recall (R), and F1. OSWorld success is determined by execution-based scripts on the final desktop state; we report both a single-run success rate and the mean over three runs.

Implementation. We use Qwen3.6-27B Qwen Team (2026) for F_{part}, F_{ref}, F_{act}, F_{grp}, and F_{rev}. We build on the open-source DART-GUI infrastructure Li et al. (2025) for reinforcement learning: multi-turn OSWorld rollouts are collected independently of training, stored, and consumed by an asynchronous GRPO trainer. Our framework for process-reward generation is integrated as an additional asynchronous stage on completed trajectories, and the trainer then uses these process rewards to shape the GRPO advantage of the base model UI-TARS-1.5-7B Qin et al. (2025). Rollouts use $N = 8$ trajectories per instruction, 15 concurrent environments, a maximum of 30 environment steps, and sampling temperature 0.7; each action is generated with at most 5 images and 35 text turns of history. GRPO training uses learning rate 1×10^{-7}, KL coefficient 0.1, clip range $[0.1, 0.28]$, and process-advantage coefficient $\alpha = 0.5$.

Baselines. On CUARewardBench, we compare against the official baselines: general vision-language models (Qwen2.5-VL Bai et al. (2025b), Qwen3-VL Bai et al. (2025a), GLM-4.5V V Team et al. (2025), and GPT-5 OpenAI (2025)) and GUI models (GUI-Owl Ye et al. (2025) and SE-WSM Sun et al. (2025)). We also run the same Qwen3.6-27B under the official templates. On CUAStepBench, we compare against CUAJudge Xue et al. (2026), OSThemis Li et al. (2026b), and

Table 3: Step-level evaluation on CUAStepBench. Best results are in bold.

Framework	Model	P	R	F1
CUAJudge	GPT-5-mini	49.7	30.8	38.0
CUAJudge	Qwen3-VL-8B	**67.7**	12.7	21.3
OSThemis	Qwen3-VL-8B	17.0	14.3	15.6
SeekJudge	Qwen3-VL-8B	41.4	20.4	27.3
SeekJudge	SeekJudge-9B	44.5	33.3	38.1
SeekJudge	Qwen3.6-27B	44.0	30.8	36.3
Ours	Qwen3.6-27B	30.6	**57.2**	**39.9**

Table 4: Task success rate (%) on OSWorld. All methods use UI-TARS-1.5-7B as the base model. Impress and Calc denote LibreOffice Impress and LibreOffice Calc. OA is the overall success rate over the three applications' tasks.

	Chrome		Impress		Calc		OA	
Method	1 run	3-run mean	1 run	3-run mean	1 run	3-run mean	1 run	3-run mean
UI-TARS	32.6	28.2	33.8	32.1	10.6	7.6	25.6	22.6
Outcome GRPO	43.5	37.0	44.7	41.6	19.1	12.8	35.7	30.4
SeekJudge	45.7	38.3	46.8	41.1	19.1	14.9	37.1	31.4
Ours	**49.9**	**43.4**	**48.7**	**43.3**	**23.4**	**17.0**	**40.6**	**34.5**

SeekJudge Wan et al. (2026), including SeekJudge with the same Qwen3.6-27B backbone. For reinforcement learning on OSWorld, we compare against the base model, outcome-only GRPO Shao et al. (2024), and SeekJudge as a process-reward alternative.

4.2 Process Reward Evaluation

Tables 1 and 2 compare the proposed framework with official CUARewardBench baselines run under the OpenCUA-Reflector and SE-WSM prompt templates, respectively. Table 3 reports precision, recall, and F1 for identifying error steps on CUAStepBench. As shown in Tables 1 and 2, our framework achieves the highest overall accuracy (81.3), 3.6 points above GPT-5 with OpenCUA-Reflector (77.7) and 6.4 points above GPT-5 with SE-WSM (74.9). Our framework also improves overall accuracy by 14.0 points over the same Qwen3.6-27B backbone under OpenCUA-Reflector (67.3) and by 10.5 points under SE-WSM (70.8), demonstrating that the gain comes from the grouping and two-level scoring rather than from a stronger judge model. The two official templates also differ: OpenCUA-Reflector gives higher recall but lower specificity, and SE-WSM gives higher specificity but lower recall (e.g., Qwen3.6-27B at 95.1/36.6 vs. 69.8/72.0; GPT-5 at 92.3/61.6 vs. 64.8/86.0), while our framework keeps a more even trade-off (87.8 recall, 73.9 specificity). We then evaluate our framework on CUAStepBench, as shown in Table 3. Our framework achieves the highest F1 (39.9), 1.8 points above SeekJudge-9B (38.1). With the same Qwen3.6-27B backbone, SeekJudge reaches only 36.3 F1, while our framework improves F1 by 3.6 points. Our framework also achieves the highest recall (57.2), at lower precision (30.6 vs. 44.0), thus missing fewer error steps.

4.3 Reinforcement Learning on OSWorld

As shown in Table 4, we train UI-TARS-1.5-7B on Chrome, LibreOffice Impress, and LibreOffice Calc. UI-TARS is the base model without RL; Outcome GRPO is standard GRPO trained with the outcome reward; SeekJudge and Ours both augment GRPO with process rewards, from SeekJudge and from our framework, respectively. Ours achieves the highest success rate on all three applications and the highest overall success rate (OA). OA is 40.6/34.5 (1-run / 3-run mean), 15.0/11.9 points above UI-TARS and 4.9/4.1 points above Outcome GRPO. This shows that process-reward advantage shaping improves over standard GRPO with the outcome reward alone. With the same shaping, Ours further outperforms SeekJudge by 3.5/3.1 OA points (40.6/34.5 vs. 37.1/31.4). Sampling throughput under synchronous and asynchronous scoring is reported in Appendix B.1.

Table 5: Grouping ablation on CUARewardBench. Single-group places all actions in one group. Fixed-window denotes the fixed-window step-reward methods, consecutive windows of 5 screenshots. Grouping w/ screenshots additionally uses screenshots when partitioning.

Grouping	P	NPV	R	S	OA
Single-group	69.1	**96.3**	**98.4**	49.7	75.7
Fixed-window	67.4	93.8	97.2	47.2	73.7
Grouping w/ screenshots	**79.4**	81.8	85.1	**75.2**	80.4
Ours	**79.4**	84.1	87.8	73.9	**81.3**

Table 6: Scoring and aggregation ablation on CUARewardBench. w/o reference omits u_k. w/o refinement omits F_{rev}. w/o group scoring uses only r_t^{act}. w/o action scoring uses only $\tilde{r}_{k(t)}^{\text{group}}$. Sum aggregation replaces the product with an equal sum of the two scores.

Scoring	P	NPV	R	S	OA
w/o reference	74.7	83.7	89.0	65.2	77.9
w/o action scoring	68.6	85.3	92.2	51.6	73.3
w/o group scoring	70.8	91.5	95.6	54.8	76.6
w/o refinement	76.0	86.2	90.7	67.1	79.7
Sum aggregation	70.6	**94.4**	**97.2**	53.5	76.9
Ours	**79.4**	84.1	87.8	**73.9**	**81.3**

4.4 Ablation Study

As shown in Table 5, we ablate grouping on CUARewardBench. Ours uses adaptive grouping from the instruction and the agent's responses. Single-group places all actions in one group. Fixed-window denotes the fixed-window step-reward methods described in section 1, using consecutive windows of 5 screenshots. Grouping w/ screenshots uses screenshots as extra input when partitioning. Ours achieves the highest overall accuracy (81.3). Single-group lowers overall accuracy to 75.7, which shows that splitting the trajectory into subgoal groups is useful because a subgoal is specific enough to tell useful actions from redundant or incorrect ones, rather than judging them against the whole task. Fixed-window lowers overall accuracy further to 73.7, since a preset local window can miss evidence outside the window or bring in context that does not belong to the executed subgoal. Grouping w/ screenshots also hurts, bringing overall accuracy down to 80.4, because screenshots can introduce interference, as visually similar screenshots may belong to different subgoals.

As shown in Table 6, we ablate scoring and aggregation on CUARewardBench. w/o reference removes u_k. w/o refinement removes F_{rev} and keeps the initial group-level scores. w/o group scoring removes F_{grp} and F_{rev} and uses only r_t^{act}. w/o action scoring removes F_{act} and uses only $\tilde{r}_{k(t)}^{\text{group}}$. Sum aggregation replaces the product with an equal sum of the two scores. Ours achieves the highest overall accuracy (81.3). Removing any scoring stage, or replacing the product with an equal sum, drops accuracy, demonstrating the effectiveness of each scoring stage and of the product aggregation.

A qualitative analysis of the proposed process rewards is given in Appendix C.

5 Conclusion

In this paper, we present an adaptive-group process reward framework that dynamically partitions a GUI trajectory into semantically coherent subgoal groups, enabling process reward estimation at both the action and group levels. The process reward of each action is obtained by combining its action-level evaluation with the refined group-level evaluation. These process rewards are used to shape GRPO's step-level advantage during policy optimization. Experiments on process reward benchmarks and downstream RL training demonstrate the effectiveness of the proposed framework.

AI USE STATEMENT

In this work, we used generative AI tools for assisting with translation, providing feedback on research methodology and experiments, and interpreting results. We have not used generative AI tools for developing theoretical models or conceptual frameworks, proposing or refining hypotheses, or implementing methods, and generating synthetic datasets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis are not applicable to this work. We have reviewed all AI-assisted work: translated text was checked against the intended meaning, feedback on methodology and experiments was treated as advisory and final design decisions were made by the authors, and interpretations of results were verified against the experimental numbers. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s: An open agentic framework that uses computers like a human. *arXiv preprint arXiv:2410.08164*, 2024.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Cong Chen, Kaixiang Ji, Hao Zhong, Muzhi Zhu, Anzhou Li, Guo Gan, Ziyuan Huang, Cheng Zou, Jiajia Liu, Jingdong Chen, Hao Chen, and Chunhua Shen. Gui-shepherd: Reliable process reward and verification for long-sequence gui tasks. *arXiv preprint arXiv:2509.23738*, 2025.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *ACL*, pp. 9313–9332, 2024.

Gaole Dai, Shiqi Jiang, Yuanchun Li, Ting Cao, Mo Li, Rui Tan, Yuqing Yang, and Lili Qiu. Prore: A proactive reward system for gui agents via reasoner–actor collaboration. *arXiv preprint arXiv:2509.21823*, 2026a.

Silong Dai, Changzhi Sun, Haolun Wu, Huanran Zheng, Tao Ji, Junchi Yan, Yuanbin Wu, Dell Zhang, Xiaoling Wang, and Xuelong Li. Group verification-based policy optimization for interactive coding agents. In *ICLR*, 2026b.

Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaojian Ma, Tao Yuan, Yue Fan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. Multi-modal agent tuning: Building a vlm-driven agent for efficient tool usage. *arXiv preprint arXiv:2412.15606*, 2024.

Haojie Hao, Longkun Hao, Yihang Lou, Yan Bai, Zhenyang Li, Zhichao Yang, Dongshuo Huang, Hongyu Lin, Lanqing Hong, Jiakai Wang, and Xianglong Liu. Stainflow: Entity-stain tracking and evidence linking for process rewards in gui agents. *arXiv preprint arXiv:2606.07027*, 2026.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *CVPR*, pp. 14281–14290, 2024.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven C. H. Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *arXiv preprint arXiv:2207.01780*, 2022.

Pengxiang Li, Zechen Hu, Zirui Shang, Jingrong Wu, Yang Liu, Hui Liu, Zhi Gao, Chenrui Shi, Bofei Zhang, Zihao Zhang, Xiaochuan Shi, Zedong Yu, Yuwei Wu, Xinxiao Wu, Yunde Jia, Liuyu Xiang, Zhaofeng He, and Qing Li. Efficient multi-turn rl for gui agents via decoupled training and adaptive data curation. *arXiv preprint arXiv:2509.23866*, 2025.

Zecheng Li, Zhihui Cao, Wenke Huang, Yudong Zhang, Keying Qi, Rui Wang, Zeyu Zheng, Jian Zhao, Hao Zhu, Hengxin Wu, Yuran Wang, Guitao Fan, Guokun Wu, Yicong Liu, Zhilin Gao, Haikun Xu, He Yang, Minqi Xiang, Xingyu Liu, and Zuojian Wang. Magicgui-rms: A multi-agent reward model system for self-evolving gui agents via automated feedback reflux. *arXiv preprint arXiv:2601.13060*, 2026a.

Zehao Li, Zhenyu Wu, Yibo Zhao, Bowen Yang, Jingjing Xie, Zhaoyang Liu, Zhoumianze Liu, Kaiming Jin, Jianze Liang, Zonglin Li, et al. Os-themis: A scalable critic framework for generalist gui rewards. *arXiv preprint arXiv:2603.19191*, 2026b.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Haojia Lin, Xiaoyu Tan, Yulei Qin, Zihan Xu, Yuchen Shi, Zongyi Li, Gang Li, Shaofei Cai, Siqi Cai, Yuzheng Cai, Chaoyou Fu, Ke Li, and Xing Sun. Cuarewardbench: A benchmark for evaluating reward models for computer-using agents. In *ICML*, 2026.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. *arXiv preprint arXiv:2411.17465*, 2024.

Jiate Liu, Yiqin Zhu, Kaiwen Xiao, Qiang Fu, Xiao Han, Wei Yang, and Deheng Ye. Rltf: Reinforcement learning from unit test feedback. *arXiv preprint arXiv:2307.04349*, 2023.

Junyu Lu, Songxin Zhang, Zejian Xie, Zhuoyang Song, and Jiaxing Zhang. Orcust: Stepwise-feedback reinforcement learning for gui agent. *arXiv preprint arXiv:2509.17917*, 2025.

OpenAI. Gpt-5 system card, 2025. URL `https://openai.com/index/gpt-5-system-card/`.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, 2026. URL `https://qwen.ai/blog?id=qwen3.6-27b`.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. Seagent: Self-evolving computer use agent with autonomous learning from experience. *arXiv preprint arXiv:2508.04700*, 2025.

Fei Tang, Zhiqiong Lu, Boxuan Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Clawgui: A unified framework for training, evaluating, and deploying gui agents. *arXiv preprint arXiv:2604.11784*, 2026.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, et al. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*, 2025.

Yang Wan, Zhenhao Zhang, Jierui Wang, and Linchao Zhu. Seekjudge: A practical reward framework for reinforcement learning in computer-use agents. *arXiv preprint arXiv:2607.23263*, 2026.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, Peihang Li, Fangyu Lei, Ruisheng Cao, Yeqiao Fu, Dongchan Shin, Martin Shin, Jiarui Hu, Yuyan Wang, Jixuan Chen, Yuxiao Ye, Danyang Zhang, Dikang Du, Hao Hu, Huarong Chen, Zaida Zhou, Yipu Wang, Heng Wang, Diyi Yang, Victor Zhong, Flood Sung, Y. Charles, Zhilin Yang, and Tao Yu. Opencua: Open foundations for computer-use agents. *arXiv preprint arXiv:2508.09123*, 2025.

Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, Shuai Ren, Yafei Wen, Xiaoxin Chen, Aojun Zhou, and Hongsheng Li. Ui-genie: A self-improving approach for iteratively boosting mllm-based mobile gui agents. *arXiv preprint arXiv:2505.21496*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Jing Hua Toh, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *NeurIPS*, volume 37, pp. 52040–52094, 2025.

Tao Xiong, Xavier Hu, Yurun Chen, Yuhang Liu, Changqiao Wu, Pengzhi Gao, Wei Liu, Jian Luan, and Shengyu Zhang. Gui-pra: Process reward agent for gui tasks. *arXiv preprint arXiv:2509.23263*, 2025.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. In *ICML*, 2025.

Tianci Xue, Zeyi Liao, Tianneng Shi, Zilu Wang, Kai Zhang, Dawn Song, Yu Su, and Huan Sun. Autonomous continual learning for environment adaptation of computer-use agents. *arXiv preprint arXiv:2602.10356*, 2026.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, et al. Mobile-agent-v3: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*, 2025.

Yuwen Zhai, Runze Li, Liang Wang, Nian Shi, Liwu Xu, Wei Zhang, Ran Lin, Bo Xu, and Benlei Cui. Guide: Interpretable gui agent evaluation via hierarchical diagnosis. *arXiv preprint arXiv:2604.04399*, 2026.

Bofei Zhang, Zirui Shang, Zhi Gao, Wang Zhang, Rui Xie, Xiaojian Ma, Tao Yuan, Xinxiao Wu, Song-Chun Zhu, and Qing Li. Tongui: Internet-scale trajectories from multimodal web tutorials for generalized gui agents. *arXiv preprint arXiv:2504.12679*, 2025.

Congmin Zheng, Xiaoyun Mo, Xinbei Ma, Qiqiang Lin, Yin Zhao, Jiachen Zhu, Xingyu Lou, Jun Wang, Zhaoxiang Wang, Weiwen Liu, Zhuosheng Zhang, Yong Yu, and Weinan Zhang. Adaptive milestone reward for gui agents. *arXiv preprint arXiv:2602.11524*, 2026.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A Additional Implementation Details

We list the score mapping M and the critic prompts for F_{part}, F_{ref}, F_{act}, F_{grp}, and F_{rev}.

A.1 Score mapping

Table 7 maps the two labels to a scalar in $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. The same table is used at both levels: e_t/e_k as execution or subgoal status, and p_t/p_k as subgoal or task contribution.

Table 7: Lookup table M mapping the two labels to a scalar. The same table is used at both the action level and the group level.

Contribution	Completed	Partial	Failed	Uncertain
Milestone	1.0	0.4	0.0	0.4
Useful-efficient	0.8	0.4	0.0	0.4
Useful-inefficient	0.6	0.4	0.0	0.4
Exploratory	0.4	0.4	0.0	0.2
Redundant	0.2	0.2	0.0	0.2
Harmful	0.0	0.0	0.0	0.0

A.2 Grouping critic prompt

F_{part} reads the instruction g and the responses $\{c_t\}$ and returns contiguous groups $G_k = (n_k, \mathcal{I}_k)$.

Grouping critic prompt

```
You are an expert grouping critic for GUI agents.
Partition the GUI trajectory into contiguous subgoal groups based on the User Instruction.

Format:
[
    {
        "subgoal_name": "Summary",
        "start_step": <int>,
        "end_step": <int>,
        "reasoning": "Why these steps are grouped"
    }
]

## User Task Instruction
{g}

## Trajectory Steps
Step 1: {c_1}
...
Step T: {c_T}

Analyze the steps and group them. The groups MUST cover every step from
Step 1 through Step {T} exactly once, in order, with no gaps or
overlaps. Include a final finished/DONE/FAIL step in the last group when
one is present.
```

A.3 Reference critic prompt

F_{ref} builds the instruction-conditioned reference $u_k = (u_k^{\text{tgt}}, u_k^{\text{cst}}, u_k^{\text{prc}})$ for group G_k.

Reference critic prompt

```
You are an expert Visual Reward Model for GUI Agents.
Build a rule-consistent task reference for later judgments.

## Global Task Instruction
```

```
{g}

## Subgoal Trace (Context):
**1. History Group Records:**
{H_<k} or N/A (No previous scored groups)

**2. Current Group Record:**
{G_k record: n_k, grouping rationale}

**3. Future Group Records:**
{later group names/rationales, or N/A}

## Current Group Step Text (Reference Only)
{c_t for t in I_k}

## Audit Instructions
- Purpose: build the reference used by all later judgments.
- Build a rule-consistent reference before judging steps or groups.
- Step text may reveal candidate objects, values, formulas, routes, or assumptions, but it
    does not prove correctness or execution.

Target state:
- The correct state required by the global task and the current group if the group is useful.
- Derive only instruction-grounded requirements from the Global Task Instruction and explicit
     task rules.
- The current subgoal name is a hypothesis from trajectory segmentation, not ground truth.
- Do not assume the current subgoal is a correct route to the global task.
- If the current subgoal follows a route that task rules already identify as wrong or
    unsupported, target_state must describe the correct-route target state, not "recognize/
    verify that the wrong route fails".
- Do not copy the agent's proposed answer from Thought/Action unless it is verified by the
    task rules.
- target_state must not contradict rules. If the agent's intended value conflicts with the
    rules, target_state must follow the rules.

Rules:
- Exact constraints needed to judge correctness: values, labels, mappings, thresholds,
    formulas, formatting, line breaks, separators, row/column/cell/field boundaries,
    ordering, and what must remain unchanged.
- Include task-level rules needed to judge whether the current subgoal is a correct,
    exploratory, or wrong route.
- For mapping/classification/threshold tasks, explicitly compute or state the rule that
    determines the expected label/value.

Preferred process:
- The correct or robust task-solving process when a scalable method or known UI/tool route is
     relevant.
- Do not require exact imitation when another process achieves the same correct target state.
- Do not describe the agent's attempted process as preferred merely because it appears in the
     trajectory.
- If the current group is a wrong-path or exploratory attempt, describe the correct route or
    state that the attempt is only diagnostic/exploratory.
- If task rules already identify a route, tool, menu, or operation as wrong or unsupported,
    preferred_process must start from the correct route instead of including that known-
    wrong attempt as a required confirmation step, unless the original user task explicitly
    asks to test that route. The agent's subgoal name, reasoning, or later failure does not
    count as a user request to test it.
- For spreadsheet/data tasks, prefer formulas, fill-down, range edits, sorting/filtering, and
     batch operations over repeated cell-by-cell edits.

## Output Format (JSON)
{
  "task_reference": {
    "target_state": "<string> Exact target state required by the task/subgoal.",
    "rules": "<string> Values, mappings, thresholds, formatting, boundaries, ordering,
     formulas, or constraints that must be checked.",
    "preferred_process": "<string> Expected high-quality/scalable process when relevant."
  }
}
```

A.4 ACTION-LEVEL SCORING CRITIC PROMPT

F_{act} scores all target actions of a sliding window inside G_k in one call. Screenshots are attached only to the window being scored; other actions in the group are given as text.

Action-level scoring critic prompt

```
You are an expert Visual Reward Model for GUI Agents.
Score the target steps of the current group.

## Global Task Instruction
{g}

## Task Reference:
{u_k = (u_k^tgt, u_k^cst, u_k^prc)}

## Target Subgoal:
Subgoal: {n_k}
Reasoning: {grouping rationale}

## Audit Instructions
- Purpose: evaluate each target step using two axes: action execution and subgoal
    contribution.
- Evaluate each target step independently, then use later steps only to judge whether its
    effect was useful.
- For finished() actions, judge whether stopping/claiming completion was appropriate.

Intent summary:
- The immediate GUI action intended by the agent, inferred from Thought/Agent Response/Action
     text.
- The agent's stated rule or category is not authoritative; verify it against the task
    reference.

Expected subgoal effect:
- What this step should accomplish according to the task/subgoal rules, independent of the
    agent's Thought.
- Derive this from task_reference.rules and visible source data; do not copy the agent's
    unverified proposed answer.
- For mapping/classification/threshold/formula/text-format tasks, compute the expected value,
     label, line break, separator, cell/field/range, order, or format from the task
    reference.

Observed effect:
- The actual UI/state/content change visible in screenshots, or why the evidence is
    insufficient.

Execution result:
- This is the low-level grounding axis. Compare intent_summary with observed_effect; do not
    judge whether the semantic purpose was correct here.
- Completed: the intended immediate GUI effect is fully supported by visual/state evidence,
    regardless of whether that intent was semantically correct or efficient; Thought alone
    is insufficient.
- Partial: only part of the intended immediate GUI effect is visibly achieved.
- Failed: no change, wrong change, error state, repeated same intent, admitted failure, or
    premature finished().
- Uncertain: visual/text evidence is insufficient to determine execution; use it only for
    grounding uncertainty, not uncertainty about semantic contribution.

Subgoal contribution:
- This is the high-level plan/strategy axis. Compare intent_summary with
    expected_subgoal_effect and task_reference to judge the purpose, method, and efficiency,
     separately from whether the GUI action executed.
- Milestone: the intended effect is the decisive operation that would complete the current
    subgoal or produce its required deliverable/end state, including a successful correction
     that directly establishes that state; preparation and navigation are not milestones.
- UsefulEfficient: the semantic purpose is correct under the task/subgoal constraints, not
    merely topically related, and usefully advances the subgoal through an appropriate,
    direct, and scalable method; this includes a necessary on-path transition, correct
    decision, or successful recovery/error correction.
- UsefulInefficient: the semantic purpose is correct and useful, including a useful
    transition or recovery, but the method is substantially manual, repetitive, brittle,
    indirect, or non-scalable; for example, filling cells individually when a reliable batch
     fill is available.
- Exploratory: the semantic purpose is reasonable information-gathering needed under genuine
    uncertainty, even when it is not direct task progress.
- Redundant: the semantic purpose is unnecessary, repetitive, meaningless, or off-path and
    provides neither useful progress nor useful information.
- Harmful: the semantic purpose is an actively wrong or harmful off-path action, false
    completion claim, ignored needed correction, or otherwise targets a wrong value, label,
    format, object, route, rule, or state.

Cross-axis rules:
- Judge the two axes independently. A semantically wrong action can be fully executed; use
    Completed + Harmful in that case.
```

```
- Partial means that the step did not fully realize its own immediate intended effect, not
    that the entire subgoal remains incomplete.

## Current Group Trace:
For each step t in the current group G_k:
- If t is a target step in the current scoring window:
  [Target Step Evidence]:
  <image of o_t>
  Step {t}: {c_t}
  <image of o_{t+1}>  (attached after the last target step in the window)
- If t is outside the current window:
  [Context Text Only]: Step {t}: {c_t}

## Output Format (JSON)
{
  "step_critic": [
    {
      "step_index": <int> (MUST be one of: [{target step indices}]),
      "is_finished_action": <bool>,
      "intent_summary": "<string> Intended action inferred from text/action.",
      "expected_subgoal_effect": "<string> What this step should accomplish according to the
     task/subgoal rules, independent of the agent's Thought.",
      "observed_effect": "<string> Actual UI/state change observed from screenshots, or why
     uncertain.",
      "execution_result": "<string> 'Completed', 'Partial', 'Failed', or 'Uncertain'.",
      "subgoal_contribution": "<string> 'Milestone', 'UsefulEfficient', 'UsefulInefficient',
     'Exploratory', 'Redundant', or 'Harmful'.",
      "reason": "<string>"
    }
  ]
}
```

A.5 GROUP-LEVEL SCORING CRITIC PROMPT

F_{grp} scores G_k from the action-level judgments J_k, the reference u_k, and preceding summaries $H_{<k}$.

Group-level scoring critic prompt

```
You are an expert Visual Reward Model for GUI Agents.
Evaluate one target subgoal group for subgoal completion and contribution to the task.

## Global Task Instruction
{g}

## Task Reference:
{u_k = (u_k^tgt, u_k^cst, u_k^prc)}

## Subgoal Trace (Context):
**1. History Group Records:**
{H_<k} or N/A (No previous scored groups)

**2. Current Group Record:**
{G_k record: n_k, grouping rationale}

**3. Future Group Records:**
{later group names/rationales, or N/A}

## Current Group Step-Level Evidence:
{J_k}

## Audit Instructions
- Purpose: evaluate the current subgoal group using two axes: subgoal status and goal
    contribution.
- Evaluate the group after reviewing all step-level judgments.

Subgoal intent summary:
- What this group is trying to accomplish, inferred from the subgoal name, reasoning, and
    actions.
- The subgoal name is not authoritative; summarize the attempted intent without assuming it
    is correct.

Expected goal effect:
```

```
- The effect this group should have on the global task after judging whether the current
    subgoal route is correct or wrong.
- If the route is correct, state the concrete task-relevant effect it should produce: correct
     page/menu/tool, correct file/content/value/layout, correct generated output, correct
    setting, correct selection, or correct progress toward the requested result.
- If the route is wrong, unsupported, unnecessary, or based on a false assumption, state that
     following this route is not expected to produce useful progress toward the global task.
- Do not convert a wrong route into a useful diagnostic objective unless the user explicitly
    asked to test or verify that route.

Observed group effect:
- The actual final UI/file/state/content effect of the group, based on the merged step-level
    judgments and their visual-evidence reasons.

Subgoal status:
- This is the group-level grounding axis. Compare subgoal_intent_summary with
    observed_group_effect, independently of whether the group intent was semantically
    correct.
- Completed: the group fully realized its intended group-level effect.
- Partial: the group realized only part of its intended group-level effect.
- Failed: the group did not realize its intended group-level effect.
- Uncertain: the available step/visual evidence is insufficient to determine group-level
    execution; do not use it for uncertainty about semantic contribution.

Goal contribution:
- This is the global plan/strategy axis. Compare subgoal_intent_summary with
    expected_goal_effect and task_reference to judge the route, method, and efficiency,
    separately from whether the group intent was realized.
- Milestone: the intended group effect is decisive for completing an important global-task
    requirement or producing its required deliverable/end state, including a successful
    correction that directly establishes that state; preparation and navigation groups are
    not milestones.
- UsefulEfficient: the group intent is correct under the task constraints, not merely
    topically related, and advances the global task through an appropriate, direct, and
    scalable route; this includes a necessary on-path transition, correct decision, or
    successful recovery/error correction.
- UsefulInefficient: the group intent is correct and useful, including a useful transition or
     recovery, but its route is substantially manual, repetitive, brittle, indirect, or non-
    scalable.
- Exploratory: the group reasonably gathers information needed under genuine uncertainty,
    even when it does not directly advance the task.
- Redundant: the group is unnecessary, repetitive, meaningless, or off-path and provides
    neither useful progress nor useful information.
- Harmful: the group follows an actively wrong or harmful route, makes a false completion
    claim, ignores a needed correction, produces wrong content/output, corrupts required
    state, or otherwise moves away from the global task.

Cross-axis rules:
- Judge the two axes independently. A semantically wrong group can fully realize its intended
     effect; use Completed + Harmful in that case.
- Partial means that the group only partly realized its own intended group-level effect, not
    that the entire global task remains incomplete.
- Base observed_group_effect on current step evidence; Previous Group Results are historical
    context and do not override it.

## Output Format (JSON)
{
  "current_group_conclusion": {
    "subgoal_intent_summary": "<string> What this group is trying to accomplish, without
     assuming the subgoal is correct.",
    "expected_goal_effect": "<string> What this group should contribute to the global task if
      it is correct/useful.",
    "observed_group_effect": "<string> Actual final UI/file/state/content effect of the group
     .",
    "subgoal_status": "<string> 'Completed', 'Partial', 'Failed', or 'Uncertain'.",
    "goal_contribution": "<string> 'Milestone', 'UsefulEfficient', 'UsefulInefficient', '
     Exploratory', 'Redundant', or 'Harmful'.",
    "reason": "<string>",
    "history_summary_for_next_group": "<string> One concise sentence stating what was
     achieved, failed, or learned for later groups."
  }
}
```

A.6 REFINEMENT CRITIC PROMPT

F_{rev} revisits all group-level labels after every group has been scored, using the full sequence of group conclusions.

Refinement critic prompt

```
You are an expert Visual Reward Model for GUI Agents.
Refine group-level judgments using full trajectory-level group evidence.

## Global Task Instruction
{g}

## Initial Group-Level Evaluations
{G_k, J_k, e_k, p_k for k=1..K}

## Audit Instructions
- Purpose: refine group-level judgments after all groups have initial step-level and group-
    level judgments.
- Refine only group-level conclusions; do not change step-level execution results.

Cross-group dependency:
- Use earlier group results to judge whether a later group depends on valid or invalid
    prerequisites.
- Use later group results to judge whether an earlier group was reused, invalidated, repaired
    , abandoned, or became a dead end.
- Do not punish a correct group only because a later unrelated group failed.

Refined subgoal status:
- Completed: the group result remains visibly/semantically achieved after considering cross-
    group evidence.
- Partial: the group made meaningful progress but later evidence shows the result was
    incomplete, weakly used, or only partly valid.
- Failed: the group produced no meaningful valid result, or later evidence shows its result
    was invalid.
- Uncertain: cross-group evidence is still insufficient to determine whether the intended
    group effect was realized.

Refined goal contribution:
- Milestone: the group is decisive for a major task requirement or required deliverable/end
    state.
- UsefulEfficient: the group follows a correct, direct, and scalable route that supports
    global task progress.
- UsefulInefficient: the group supports global task progress through a correct but
    substantially inefficient route.
- Exploratory: the group produced concrete diagnostic information that later helped choose a
    better route.
- Redundant: the group was harmless but not meaningfully used or needed.
- Harmful: the group followed a wrong route, produced wrong content/value/output, corrupted
    required state, blocked progress, or caused later groups to build on invalid state.

## Output Format (JSON)
{
  "refined_group_conclusions": [
    {
      "group_index": <int> (MUST be one of: [{1,...,K}]),
      "refined_subgoal_status": "<string> 'Completed', 'Partial', 'Failed', or 'Uncertain'.",
      "refined_goal_contribution": "<string> 'Milestone', 'UsefulEfficient', '
    UsefulInefficient', 'Exploratory', 'Redundant', or 'Harmful'.",
      "reason": "<string>"
    }
  ]
}
```

B Additional Experiments

B.1 Efficiency Analysis

As shown in Table 8, we compare sampling throughput during OSWorld RL with and without our framework, under synchronous and asynchronous scoring. Under the synchronous setting, the next trajectory starts only after process-reward scoring of the current one has finished. Under the asynchronous setting, the next trajectory starts immediately, and the completed trajectory is scored in the background. Sampling without our framework reaches 92.3 trajectories per hour under both settings. Directly adding our framework in the synchronous setting drops throughput to 60.3, whereas our asynchronous setting only slightly reduces it to 87.9.

Table 8: Sampling throughput (trajectories per hour). Sync starts the next trajectory after scoring finishes; Async starts the next immediately and scores the completed trajectory in the background. Δ is the change from adding Ours.

Setting	Sync	Async
w/o Ours	**92.3**	**92.3**
Ours	60.3	87.9
Δ	-32.0	-4.4

C Qualitative Analysis

C.1 Error Analysis

We analyze 63 critic error steps on CUARewardBench, which fall into four categories (Table 9). Insufficient application knowledge (41.3%) is the largest share: the critic does not know the correct menu, setting, syntax, or operator, so its reference treats a wrong path as the intended procedure (e.g., Password Manager instead of cookie settings, Insert-Comment instead of View-Notes, or a Calc sheet reference written with "!" rather than "."). Visual or interaction misreads (22.2%) are screenshot-level mistakes: a later date read as earlier, a filter chip taken as on, a slider taken as max, or a click that did not register. Missed instruction constraints (19.0%) are cases where g already specifies the requirement—working directory, numeric target, time range, "on sale"—but the critic scores the step without enforcing it. Overestimated group contribution (17.5%): the group-level score still treats a subgoal as helping the task when it does not (e.g., a group that resizes one layer when the task is to resize the whole image).

Table 9: Underlying causes of critic–human disagreements on CUARewardBench key steps.

Cause	Count	Share
Insufficient application knowledge	26	41.3%
Visual or interaction misread	14	22.2%
Missed instruction constraint	12	19.0%
Overestimated group contribution	11	17.5%

C.2 Case Studies

Figures 3, 4, and 5 visualize the adaptive-group process rewards on three CUARewardBench trajectories. Each cell shows the screenshot, the action, and the action-level score. A colored box groups consecutive steps of the same subgoal and shows the refined group-level score and contribution.

Figure 3 never turns on Do Not Track, but Groups 1–2 ("Navigate to Chrome Settings" and "Navigate to Privacy and Security Section") still score opening Settings and Privacy and security as useful, instead of giving every step a zero because the trajectory fails. In Group 3, clicking "Site settings" is exploratory and the back arrow is Useful-efficient (it undoes the wrong page); Group 4 scores "Privacy guide" the same way, then lowers that group to redundant because Group 3 has already shown the setting is not there. Group 5 scores going back to "Site settings" as useful, but the premature `DONE` as 0.

Figure 4 fills the column only after Group 2 ("Attempt to auto-fill sequence (First Attempt)") fails. If only the final outcome were used, the misclick that opens Activities would get the same success credit as the later fill. Grouping puts typing "No. 1" and "No. 2" in Group 1 ("Enter initial sequence values"), the failed fill in Group 2, and the retry in Group 3 ("Retry auto-fill sequence (Second Attempt)"), so the successful retry does not change the scores of the failed attempt. In Group 2, the click that opens Activities and the missed fill handle are harmful, Esc is Useful-efficient (it closes Activities and returns to Calc), and the group stays exploratory because the column is still empty; Group 3 then scores the same fill as useful after "No. #" appears.

Figure 5 shows a failure case: the process reward is wrong at the dashed step. The instruction asks for the latest email in the Bills folder. The list shows an AWS invoice dated $3/1/2024$ above an

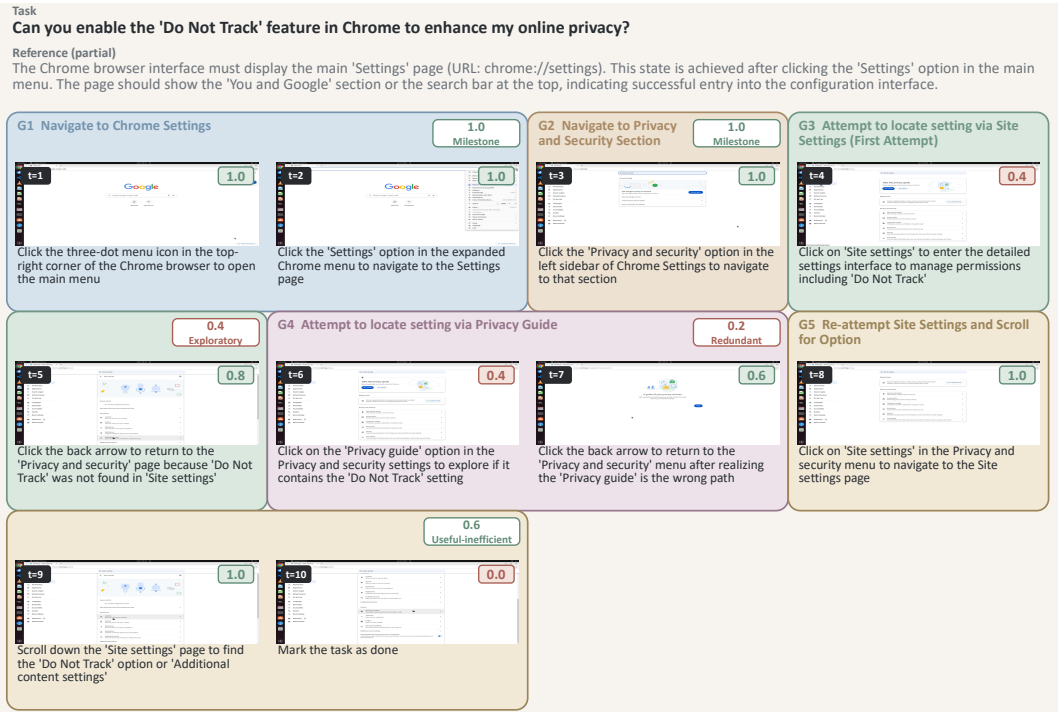

Figure 3: A Chrome trajectory for enabling Do Not Track.

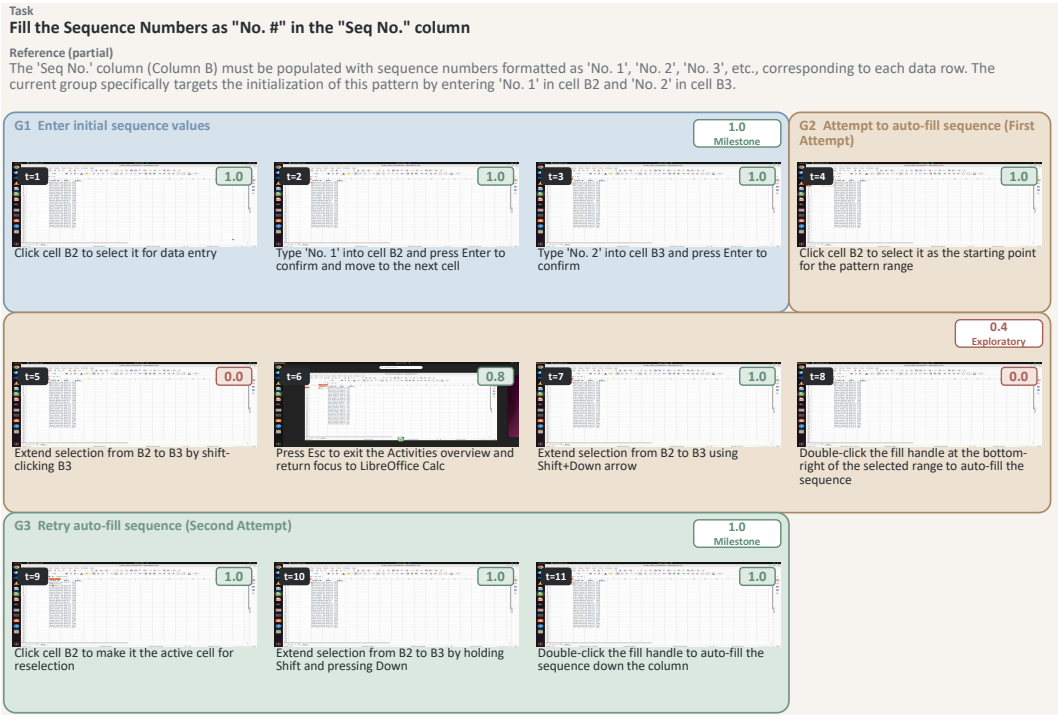

Figure 4: A Calc trajectory that fills the Seq No. column with “No. #”.

X receipt dated 12/1/2024; 12/1/2024 is later, so the second mail should be opened. The critic compares the two dates incorrectly, treats 3/1/2024 as newer, and scores clicking the AWS invoice

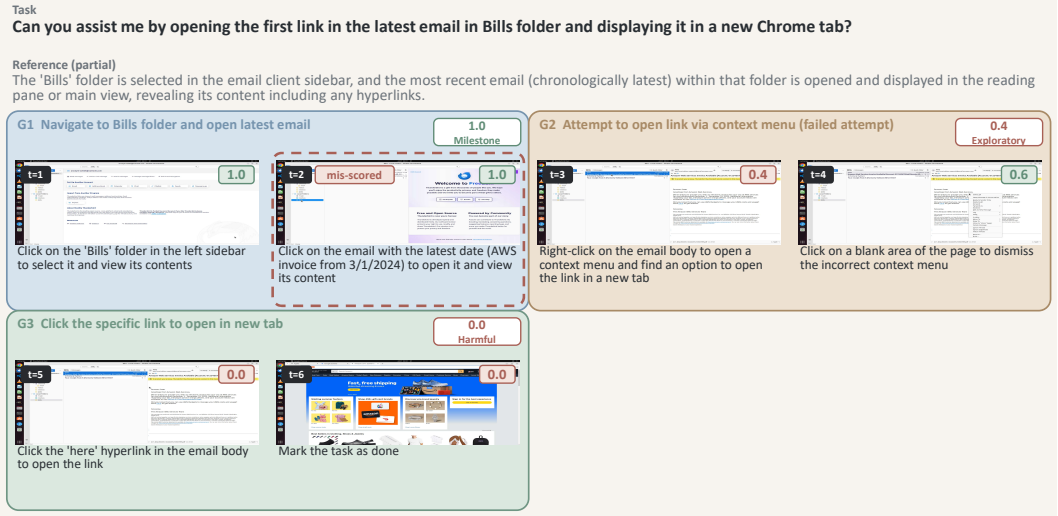

Figure 5: A failure-case trajectory that opens the first link in the latest Bills email in a new Chrome tab. The dashed outline marks the incorrectly scored step.

as 1.0. This is a visual-judgment error limited by the capability of the model used as the critic: it is not reliable enough at reading a small UI detail such as which of two dates is later.